AGGREGATE ASSEMBLY PROCESS PLANNING FOR CONCURRENT ENGINEERING

by

Alima Laguda

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

University of Durham

School of Engineering

2002

University of Durham

Abstract

AGGREGATE ASSEMBLY PROCESS PLANNING FOR CONCURRENT ENGINEERING

by Alima Laguda

Doctor of Philosophy
Submitted December 2001

In today's consumer and economic climate, manufacturers are finding it increasingly difficult to produce finished products with increased functionality whilst fulfilling the aesthetic requirements of the consumer. To remain competitive, manufacturers must always look for ways to meet the faster, better, and cheaper mantra of today's economy.

The ability for any industry to mirror the ideal world, where the design, manufacturing, and assembly process of a product would be perfected before it is put into production, will undoubtedly save a great deal of time and money. This thesis introduces the concept of aggregate assembly process planning for the conceptual stages of design, with the aim of providing the methodology behind such an environment.

The methodology is based on an aggregate product model and a connectivity model. Together, they encompass all the requirements needed to fully describe a product in terms of its assembly processes, providing a suitable means for generating assembly sequences. Two general-purpose heuristics methods namely, simulated annealing and genetic algorithms are used for the optimisation of assembly sequences generated, and the loading of the optimal assembly sequences on to workstations, generating an optimal assembly process plan for any given product.

The main novelty of this work is in the mapping of the optimisation methods to the issue of assembly sequence generation and line balancing. This includes the formulation of the objective functions for optimising assembly sequences and resource loading. Also novel to this work is the derivation of standard part assembly methodologies, used to establish and estimate functional times for standard assembly operations.

The method is demonstrated using CAPABLE*Assembly*; a suite of interlinked modules that generates a pool of optimised assembly process plans using the concepts above. A total of nine industrial products have been modelled, four of which are the conceptual product models. The process plans generated to date have been tested on industrial assembly lines and in some cases yield an increase in the production rate.

TABLE OF CONTENTS

I confirm that no part of the material offered has previously been submitted by me for a degree in this or in any other University. If material has been generated through joint work, my independent contribution has been clearly indicated. In all cases material from the work of others has been acknowledged and quotations and paraphrases suitably indicated.

If I have seen further it is by standing on the shoulders of giants.
Isaac Newton, 1676

Notation

δ		Index of work relatedness, measure of sequenced operations on an assembly line
σ		Station variation index, measure of work distribution on an assembly line
α	(°)	Angle through which a part must be rotated about an axis perpendicular to the axis of insertion to repeat its orientation.
β	(°)	Angle through which a part must be rotated about the axis of insertion to repeat its orientation
$\mathfrak{R}$		Defines an allele set
c		Overall assembly variable for minimising assembly time for a given assembly sequence
c_1		Assembly control variable for minimising number of reorientations for a given assembly sequence
c_2		Assembly control variable for maximising parallelism for a given assembly sequence
c_3		Assembly control variable for maximising the stability of intermediate subassemblies for a given assembly sequence
C_{BT}	(£)	Transportation cost per batch
C_m	(£)	Material cost per unit production
C_{no}	(£)	Non operation cost (overhead, transportation, internal handling and storage cost)
C_o	(£)	Cost rate per operator including overheads
C_{pc}	(£)	Total assembly cost per part/product
cr		Cooling rate, this represents the rate of change of temperature with increasing number of cooling schedules
cs		Cooling schedule, this represents a finite time implementation for the simulated annealing algorithm
d	(s)	Balance delay (measure of inefficiency) in assembly line balancing

E	*(%)*	Assembly line efficiency
f		Initial assembly sequence
g		New assembly sequence generated from the initial assembly sequence, in the neighbourhood of the initial assembly sequence
i		Subscript used to identify the workstation in assembly line balancing
j		Subscript used to identify the assembly task in line balancing
m		Number of assembly tasks
Mcs		Maximum number of cooling schedules
n		Number of workstations in an assembly line
n_{ej}		Number of workstations, using a T_{ej} as cycle time
n_{hf}		Number of tasks with their task times greater than half T_c
n_{min}		Theoretical minimum number of workstations
n_Q		Number of batches
NSol		Number of new assembly sequences generated and accepted during simulated annealing process, per cooling schedule
o_{ij}		Assembly task i assigned to workstation j
PA		Overall parallelism value for a given assembly sequence
p_c		Crossover rate
pop_size		Population size
R_c	*(s^{-1})*	Given production rate
RE		Overall reorientation value for a given assembly sequence
r_{st}		Stability rating index
ST		Overall stability value for a given assembly sequence
T		Initial temperature used for the simulated annealing process
$T_{\alpha+\beta}$	*(s)*	Mean handling time for the α+β range in question
T_c	*(s)*	Ideal or theoretical cycle time obtained using R_p

T_{ej}	*(s)*	Minimum cycle time, highest T_k value
T_i	*(s)*	Assembly task time for task *i*
$T_{j,j+1}$	*(s)*	Transfer time between workstations
T_{max}	*(s)*	Mean handling time for $\alpha+\beta = 720°$
T_{min}	*(s)*	Mean handling time for $\alpha+\beta < 360°$
T_p	*(s)*	Average production time per unit product/part
T_s	*(s)*	Sum of element times at a workstation on an assembly line
TSol		Total number of assembly sequences generated during the simulated annealing process per cooling schedule
T_{wc}	*(s)*	Total assembly time on an assembly line
w		Weight attached to an optimisation process
w_{pa}		Weighting factor for the effect of parallelism on a given assembly sequence
w_{re}		Weighting factor for the effect of reorientation on a given assembly sequence
w_{st}		Weighting factor for the effect of stability on a given assembly sequence
x_{pa}		Parallelism index
x_{re}		Reorientation index
X_{re}		Maximum possible value of RE for a given assembly sequence
x_{st}		Stability index
X_{st}		Maximum possible value of ST for a given assembly sequence

1 Introduction

1.1 Background

Makers of manufactured goods have faced a multitude of challenges over the past 20 years. Those challenges include increased competition in the worldwide markets, demand for more complex goods and increasing pressure from consumers for more variety and options in the goods that are produced. To remain competitive manufacturers, must always look for ways to meet the faster, better, and cheaper mantra of today's economy. In order to do that, manufacturers are looking for new ideas, new designs and new ways of doing things. To ride both the economic and product cycles smoothly, the trick is to manage the flow of such products so that offerings in the market-place always remain fresh, without wasting profits on excessive investment on updates or re-designs. Increasingly, manufacturers are forced to optimise and enhance assembly operations and minimise product cycle-time and cost.

At the same time social and economic changes in our societies and international businesses have significantly changed the way in which production technologies are being used. Instead of the previously dominant functionally-organised factories operating in relative isolation from the market situation, manufacturing systems have become more product oriented, aiming at decreased lead times, minimal work-in-process, just-in-time flow of material, and high efficiency and flexibility of manufacturing capacity utilisation. Such trends have led to concepts such as agile manufacturing, concurrent life-cycle engineering, and most recently collaborative engineering.

The average percentage of production cost attributed to assembly is quoted to be 20% (Brown et al, 1996). Of this 20% approximately half (9.7%) has been attributed to intermediate and final assembly operations, whilst the remaining half (10.3%) is attributed to set-up and other assembly support functions such as material handling and transportation. In a bid to reduce such assembly cost, a vast amount of research has gone into the optimisation of assembly processes.

The design of a product for ease of assembly forces the integration of product and process design, thereby creating an environment referred to as concurrent engineering.

1.2 Assembly

Assembly has two features that make it especially important. Firstly it is inherently integrative; it is a bringing together of parts, and therefore all sectors involved in the

product's lifecycle. Secondly, assembly is the moment when the product comes to life, since a single part does not perform any functions by itself.

Assembly can be regarded at two levels; the large and small (Delchambre, 1996). Assembly in the small deals with the details of mating regions on parts, and the physics of joining them. Assembly in the large scale deals with logical, logistical, financial and operational issues of making products from parts. Although this research also deals with assembly in the "small" sense, the focal point is based on assembly in the "large" sense. The work presented herein addresses the issue of incorporating product assembly and assembly process planning requirements at the earliest possible stage of design. Such considerations have proven to have greater impact on reducing assembly time and cost.

For example, the car industry makes impressive claims about the benefits of such computer aided engineering tools. Chrysler, which has used both Tecnomatrix[1] and Dassault's[2] tools, says that just over a decade ago it took five years to get from the concept to the production of a new car; by the late 1990's, this time had been reduced to less than half.

It is hoped that this research will underline the importance of continued research in a key core of assembly support technologies and activities including assembly and process planning.

1.2.1 A brief history

The evolution of assembly planning, as we know it today began in the late 1960s to coincide with the advent of robots, and the possibility of robotic assembly. However, the birth of assembly planning can be dated back to the mid-eighteenth and nineteenth century, to the works of Taylor and Gilbert. They gave rise to the concept of work measurements, which is defined as the dissemination of work operations to basic body motions, the elimination of unnecessary movements, and the association of times to basic body motions performed. Maynard Cooperation eventually introduced their ideas to industry in the 1950s, in the form of Method-Time Measurements (Maynard, Stegmerten, and Schwab, 1948).

The inherent limitations of robots with regards to their dexterity, and artificial intelligence, paved the way for the onset of what is now termed assembly planning. Assembly planning is viewed as the explicit definition of processes required when

[1] Tecnomatrix Technologies, a company based in Herzlia Israel, they produce a suite of programs to simulate factories.

[2] Dassault Systèmes, a French software company that produces computer aided engineering software such as CATIA and DMAPS.

building a product from a collection of individual components. These processes include product structuring, tool selection, layout generation, and assembly time and cost estimation and/or evaluation.

Assembly had to be planned down to the last detail in order for robotic assemblies to be successful. Early attempts to accomplish this merely resulted in failure, and served only to reveal how little was known or understood about assembly. Progress was eventually made in the 1970s, spurred by the arrival of the computer age. Robotic programming, computer aided design (CAD) models of parts, assembly constraints, machine vision, the physics of part mating, and the understanding of other aspects of individual part assembly actions began their evolution during this period. Design for assembly (DFA) was also born during this decade. DFA is the systematic analysis of assembly procedure used during the early stages of product design for assessing assembly difficulties and estimating assembly time and/or cost (Dewhurst, 2001).

In the 1980s, the conception of Concurrent Engineering (CE) was born, arguably the most promising approach to improve the product process time, cost of the product, and quality of the product. CE is the establishment of cross-functional teams, which encompass the knowledge and expertise necessary to ensure that all of the product's requirements are addressed. These requirements are fashioned to ensure (1) the product meets the customer performance requirements, and (2) the product should be efficient to manufacture in order to meet cost targets.

The pressure to combine functional, and manufacturing constraints, led to the recognition that more information was needed to help designers realise the full potential of CE. This knowledge is needed mostly at the conceptual level of the design process, where the product's architecture is determined, and the corresponding production systems are planned. The 1980s saw a great increase in the capabilities of computers as well as the software to support CAD for product design, particularly solid models. In the late 80s, a number of these threads came together in the form of assembly sequence analysis, and feature-based design applied to assembly modelling.

In the 1990s, assembly planning took a further step, once again supported by the ever-increasing computing power and multimedia growth, with the introduction of artificial intelligence (AI) for assembly planning and optimisation purposes, combining the off-line activities of design for assembly with the on-line activities of scheduling and operating the assembly line. The dawn of internet/intranet technology, and virtual reality has presented a new dimension for assembly design and assembly process

planning. Such advancements have paved the way for more creative mediums for real time analysis, and support tools for the product development process. The continued evolution of such technologies has taken, and will continue to take, assembly design and planning into new dimensions.

1.3 Current trends

It is safe to assume that software systems for design, production engineering, and manufacturing in the future production paradigm will be based on four types of related models (Mäntylä and Shah, 1995):

1. Generic product knowledge: that can record generic information of products and form a repository of basic engineering and performance information. The knowledge should be systemized, dependable, available, understandable and verifiable.

2. Product models: that can represent all relevant aspects of a specific product to be manufactured while avoiding the harmful over-specification of relevant details. In particular, it must be possible to represent incomplete or vague model data when appropriate.

3. Generic process models: that can record the generic characteristics of manufacturing processes in a systematic form, including the resource needs, capability, lead time, and capacity of the process. More generally, the full range of activities in the customer order delivery process should be covered in process models.

4. Factory models: that record a collection of particular instances of particular processes that constitute a particular factory. A factory model also represents the dynamic state of the manufacturing system. Similar to the above item, the full chain of processes constituting a complete customer order delivery process may be required.

A process planning methodology that currently utilises the relational models suggested above is the concept of Aggregate process planning, developed by Maropoulos (1995). The concept of aggregate process planning is described in the following section.

1.3.1 Process Planning

Over the last 20 years there has been a move towards a more flexible approach to manufacturing. Process planning has been defined "as the subsystem responsible for the conversion of design data to work instructions, and as translating part design

specifications into the manufacturing operations required to convert a part from a rough to a finished state" (Evershim, Spur and Weil, 1991).

According to Maropoulos (1995), "process planning deals with the processes required for generating the final shape, configuration or structure of a product when starting from a given initial stage". It is a mixture of complex and interrelated tasks, which are accomplished by using suitable forms of process technology and geometric reasoning. A number of process planning problems can be solved analytically, and in those cases a solution is obtained by applying algorithms and technology constraints. In other cases, problems can be formulated analytically, but the number of resulting solutions (search space) can be very large. Solutions that cannot be derived analytically can be solved using knowledge already available within that particular domain. Both analytical and knowledge based systems are suitable for performing various process planning tasks (Maropoulos, 1995).

Many working and prototype systems exist that offer good geometric capability and their operation is frequently based on the definition of "features". Other systems use rule and knowledge-based techniques, decomposition methods, interference mechanisms and genetic algorithms. Although currently limited to simple part geometry, an investment in commercially available computer aided process planning systems (fully automated) has provided returns of up to 500% (Hayes and Wright, 1989).

According to Maropoulos, Bradley and Yao (1998) in most manufacturing companies process planning occurs after the product design stage and prior to the production control activity. As such, it is detached from and out of step with both the design and production control stages. Despite the fact that it is now widely accepted that a large percentage of the product cost is/can be accounted for during the conceptual stage, the basic anomaly of commercial process planning systems is that they are mostly used when the design has almost been completed. There is an inherent lack of concurrency between process planning and production control. This is despite the fact that production control is based on the production routes devised by the process planning activity (Maropoulos, Bradley and Yao, 1998).

The realisation of the close relationship between design, process planning and production control instigated the development of a new process planning architecture suitable for the concurrent engineering practice (Maropoulos, 1995). Details of the architecture of the recently proposed process planning system are shown on Figure 1-1.

The proposed process planning architecture consists of three main planning levels namely Aggregate, Management, and Detailed (AMD). Of particular interest is the Aggregate process planning level.

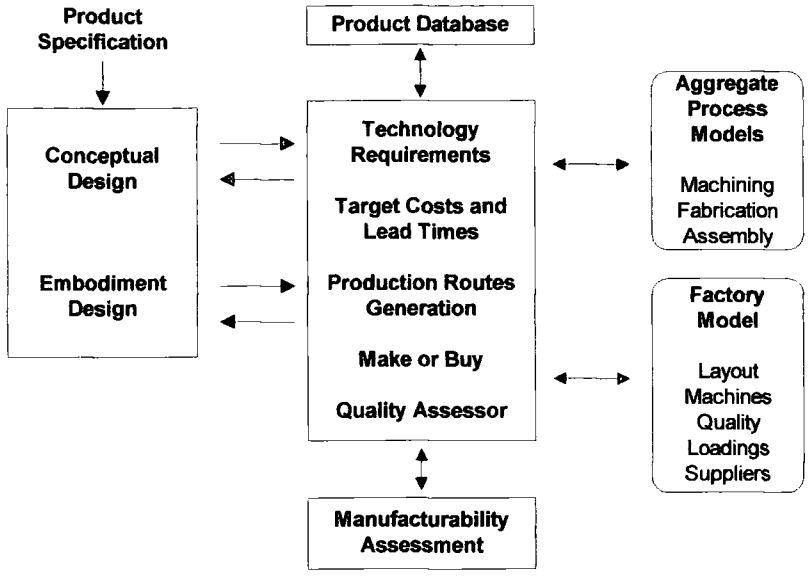


Figure 1-1 - Overall Architecture of Aggregate Process Planning (Maropoulos, 1995)

1.3.2 Aggregate Process Planning

Aggregate process planning refers to the rapid conversion of initial designs into rough cut manufacturing and assembly plans and the automatic creation of production routes. Maropoulos defines the main objectives of the planning system as follows (Maropoulos, Bradley and Yao, 1998):

1. Early evaluation and elimination of design constraints.
2. Rapid evaluation of alternative design configurations.
3. Process options and production routes and the creation of performance measures using quality cost and delivery criteria.

This functionality requires a limited amount of information concerning product and processes (see Figure 1-1) so that planning can start as early in the development cycle as possible since then there is a wide choice of options both in terms of product configuration and process selection.

Aggregate process planning is at the top level of the AMD structure and involves the planning of activities to comply with the product design cycle from the initial concept to the embodiment design. In terms of controlling cost, product development time and quality, this top level is potentially the most important level of process planning. The new technologies at this level are the generic and aggregate process and facility modelling techniques which will support the conceptual and embodiment design stages

by providing manufacturing and assembly feedback with regard to various product configurations, and assist in setting realistic cost and lead time targets (Maropoulos, Bradley, and Yao, 1998).

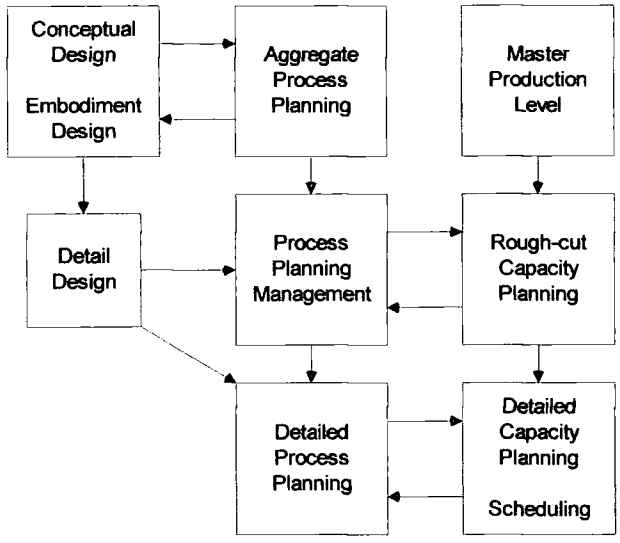


Figure 1-2 - Aggregate Process Planning Architecture (Maropoulos, Bradley and Yao, 1998)

Essential principles, shown in Figure 1-2, that govern the specification of the new aggregate process modelling methodology include:

1. Controlled simplification of detailed process models: Aggregate process models are obtained by the controlled simplification of detailed process models so that they can operate using the limited product information available during the conceptual and embodiment design stages. Any loss in accuracy is outweighed by the ability to rapidly evaluate alternate product configurations and processing options at an early design stage so that the most suitable option can be adopted.
2. Limited input data requirements: Aggregate process models are configured to function using the minimum amount of readily available product, process and factory related information. Characteristic trends between data and relationships between different types of data are vitally important at the aggregate level in order to establish a basis that will allow a comprehensive comparison of processing options and facilitate realistic decision making.
3. Utilise company specific product and process knowledge: in most cases there are opportunities to capture company specific product and process knowledge. Such information, together with specific material and process information, should be incorporated within aggregate process models.

4. Perform core technological checks concerning processes: the capability of individual processes, production equipment, manufacturing cells and of the manufacturing facility as a whole should be incorporated within aggregate process models. Only essential, core constraints should be considered at this stage and the main objectives should be to assess the generic suitability of processes, confirm the capability of specific equipment for producing certain parts and produce quantifiable measures of manufacturing performance at equipment, cell, and factory levels.
5. Incorporate only the basic, overall geometry of parts and features: the basic, overall description of parts or features should be a sufficient geometric representation of aggregate process models. The operation of aggregate process models using the minimum amount of geometrical information is vitally important so that they can be used to support the early stages of design, including conceptual design.
6. Function-Driven Operation: the members of a concurrent engineering team, mainly design and manufacturing engineers, should have equal access to the modelling methods and the corresponding process planning system.
7. Conformance with Standards: wherever possible the new modelling methods should be compatible with existing international standards concerning product design, quality and product modelling.
8. Conformance with the team-based approach: the new methods will provide rapid decision support with respect to several aspects of product development which currently require time consuming processing of information. In that respect aggregate process models improve the operational inter-disciplinary, product-based teams.

1.4 Research objectives

The initial work on aggregate process planning for assembly was undertaken by Betteridge (2000). His work included the design, and development of the aggregate product model for assembly process planning, it also contains elements of sequence generation and resource loading. The work presented herein, builds on this work. The main objective of this research is to derive a methodology to aid the generation of *optimal assembly planning process* during the *early* design stage for a given *product.*

Here, product refers to a part in its assembled state, capable of performing a predefined function. The phrase 'optimal assembly planning process' as stated above refers to the process of identifying the best possible approach to developing/building a product or part, with the ultimate aim of reducing assembly times, and thus assembly cost of the part based on available resources. The assembly planning process provides a link between the early stages of design and assembly processes and methods. This includes the generation of a suitable product model and the subsequent extraction of possible assembly planning decisions from limited part information, such as the order in which assembly operations are likely to be performed. Also, the ability to create and compare alternative product designs and configurations based on different assembly methods, processes, and resources is viewed an essential part of the assembly planning process.

Exactly how early during the design process an assembly process planning system as described above is implemented depends on the structure and nature of the company building the part. Ideally the system should be implemented at the conceptual stages of design, when a number of design configurations are being contemplated. However, if the conceptual stage solely entails the creation of abstract forms of the product, typically there is insufficient information to ascertain possible assembly features, and thus assembly processes. In such cases the assembly process planner can be implemented at the embodiment and/or detailed stages of design when an increasing number of features are being added to the existing designs.

A large number of product designs performed within manufacturing industries today involve the enhancement of a previous product design. Here, there is sufficient product data for the use of an optimal assembly process planner during the 'conceptual' stages of design. This is achieved by:

1. Developing a suitable means of representing a product model for assembly representation and sequencing. The aggregate product model is insufficient for the effective generation of feasible assembly sequences. To achieve this, a relational model is required.
2. Standardising parts and assembly operations. This is required for rapid modelling of products and processes. It also allows for the formulation of standard times for assembly operations.
3. Deriving an effective and accurate means of estimating realistic assembly operation times.

4. Generating optimal assembly sequences while satisfying assembly constraints. The assembly constraints are used to guarantee the feasibility of the sequences generated.
5. Generating optimal assembly process plans by mapping an optimised assembly sequences to a predefined or undefined factory model.

1.5 Thesis structure

The following chapter presents a detailed analysis of methodologies available to aid design for assembly. A new system for generating optimal assembly process plans is subsequently presented; CAPABLE*Assembly*.

Chapter 3 navigates the reader through the modules within CAPABLE*Assembly* prior to presenting the main methodologies that constitute CAPABLE*Assembly* in chapters 4 to 6 inclusive.

The data stored within CAPABLE*Assembly* and the methods used to develop it have been tested and validated, the documentation for these procedures is presented in Chapter 7.

A discussion on the effectiveness of CAPABLE*Assembly* as an assembly process planner, and the conclusions that can be drawn from this research are presented in Chapter 8. Finally, the recommendations for future work are also provided Chapter 8. These recommendations are of importance if CAPABLE*Assembly* is to realise its full potential.

2 Literature review

2.1 Introduction

The highly competitive nature of today's global market coupled, with the dynamic requirements of the consumer, drives the need within the manufacturing industry to decrease the life-cycle of new and/or redesigned products. At the same time, these factors also propagate the increase in complexity of the assembled products by their enhancing personalised features. In order to shorten the time required for the development of a product and its associated manufacturing processes in a concurrent engineering environment, it is desirable for the process planning activity to be automated. Furthermore, it is imperative that such process planning should be available at the earlier stages of design, for the full potential to be realised. The ability for any industry to simulate production in an ideal world, where the design, manufacturing, and assembly processes would be perfected before carrying them out for real, will undoubtedly save a great deal of time and money. The aerospace and automotive industries, for example both attribute their success in cutting the production time from concept to product by half within a decade to the use of such computer aided engineering tools.

This chapter investigates the various methodologies that have been used in attempts to optimise assembly plans at the conceptual stages of design, where product design and assembly process planning are performed in parallel and the evaluation of a design configuration is influenced by the performance of its related assembly plan.

2.2 Concurrent engineering

The discerning factor that has distinguished successful world-class companies from their competitors in recent years is the way they have managed their design process to produce manufactured goods of high quality, quickly, and on time. It is only until recently that the majority of industrial firms have acknowledged that the most effective way to manage the overall lifecycle cost of a product is through better concept design rather than innovative institutional structures and procedural competence. Hence, manufacturing companies are increasingly turning to strategic initiatives such as concurrent engineering (concurrent design and manufacturing), total quality control and just-in-time manufacturing.

Pahl and Beitz' (1984) visualisation of the philosophy of concurrent design and manufacturing is to consider all aspects of product development, during the various

stages of design, in order to avoid costly and time-consuming correcting activities downstream associated with design or manufacturing constraints. The objectives of concurrent engineering are generally agreed to be (Delchambre, 1996):

1. Improving product quality; the extent to which a product satisfies customer requirements.
2. Reducing lead times; reducing the time from product conception to successfully bringing the product to market, that is "time to market".
3. Reducing product cost; where product cost can be defined as the level of resources required to take the product from concept to market. This includes the hours worked on the product, materials used in the product and any equipment or services that are used.

The use of computer aided concurrent engineering tools can be divided into two schools of thought that need to coexist in harmony in order to establish a successful production ethos. There are those that seek to improve the communication between organisational structures through teamwork, leadership, and customer understanding and those that promote the consideration of manufacturing, quality, and assembly criteria throughout the design process.

Of particular interest to this study is the latter; the majority of tools adopted here fall under a suite of effective design techniques in product development. Examples include design for manufacturing, design for assembly, design for reliability, design for serviceability, and so on. Syan and Menon (1994) and Huang (1996) provide a comprehensive discussion of the "Design for X" philosophy. Among them, design for assembly has been applied in industries with most impressive achievements. The average percentage of production cost attributed to assembly is quoted to be 20% (Brown, et al. 1996). Other works estimate the cost of assembly to be as much as 40% to 60% for complex products (Boothroyd and Fairfield, 1991). It follows that substantial decreases in assembly cost will not only cause a visible reduction in product cost, but will further serve to shrink the product time to market and can only be seen as advantageous to any company within a competitive market.

2.3 Design for Assembly (DFA)

Automatic assembly design and planning has been recognised as an important tool for achieving concurrent product and process development thus reducing manufacturing cost. Research workers have focused on a number of aspects of design and planning for

assembly. Different approaches have been taken to develop specific parts of the overall design and planning process, these include; product and process design methodologies, computer aided design and manufacturing (CAD/CAM), design for manufacture (DFM), design for assembly (DFA) and computer simulation of assembly processes to name a few.

To a design engineer the need for an efficient DFA method is two-fold. The first need is where a designer is considering the conceptual design of a product and is attempting to determine the trade-off between assemblies containing several parts secured together or fewer parts manufactured by forging or casting. The second need is where the part has been identified and considerations are being given to the various alternative combinations of manufacturing processes and material that might be employed for its production.

The design of products, tools and processes for ease of assembly is essential if a reduction in assembly cost and an increase in the effectiveness of assembly operations are to be realised. Experiences with cost models have shown that when product designers are given the tools necessary for making early cost estimates and trade-off decisions (Boothroyd and Alting, 1992), considerable savings in subsequent manufacturing and assembly cost are possible.

Techniques and methodologies to analyse products for ease of manufacture are increasingly being adopted. Currently, the most popular DFA method is the Boothroyd Dewhurst approach (Boothroyd and Reynolds, 1989). The Boothroyd and Dewhurst approach is discussed in the following section, in their approach a product is analysed with regards to various "ease of assembly" criteria such as part symmetry and mating direction. In the original DFA method as devised by Boothroyd and Dewhurst, estimates of assembly times are based on a group technology approach where those design features of parts and products that affect assembly times are classified into broad categories. For each category an average handling and insertion time estimate is established. Clearly, for any particular assembly operation these average times can be considerably higher or lower than the actual times. However, for assemblies containing a significant number of parts the differences tends to cancel so that the resulting total time is reasonably accurate.

Other design for assembly methods includes the Hitachi Assembly Evaluation Method (AEM) method and the Lucas method. The AEM approach is based on the principle of "one motion for one part" (Miyakawa and Ohashi, 1986). The methods consist of

approximately 20 symbols that are used to represent assembly operations. Each symbol has an index, which can be used to assess the assemblability of the part under consideration. The advantages reaped from the use of this method include a reduction in assembly labour, facilitation of factory automation, reduction in design period and improved reliability of products and automated equipment (Boothroyd and Alting, 1992).

The Lucas method consists of three steps; firstly, a functional analysis of the part is performed, then a handling and feeding analysis is carried out on the parts, and finally a fitting analysis based on the assembly sequence is executed.

2.3.1 Boothroyd and Dewhurst Design For Assembly method

Since 1977 analytical methods have been developed for determining the most cost effective assembly process for a product. The Boothroyd and Dewhurst (Boothroyd and Dewhurst, 1987) DFA method is one such method. Boothroyd and Dewhurst consider assembly to include all actions including part acquisition, part insertion and the securing of the parts. The conventional definition refers to assembly plainly in terms of insertion and fastening.

According to Boothroyd, the DFA method is defined as a structured designs analysis method, which guides the design team towards a robust and elegant product structure. Indeed, there have been many published examples of successes obtained by using the Boothroyd and Dewhurst DFA method. The method was first introduced in handbook form in 1980 and then subsequently as software packages in 1982. As it exists today the method can be applied as a means by which the number of parts of a product can be reduced. In a secondary mode, it can also be used for the calculation of assembly times and identifying difficulties, which may hinder the manufacturing process or affect the quality of the product. Of particular interest is the use of the Boothroyd Dewhurst method to calculate estimated assembly times.

The time standards (Appendix A) developed by Boothroyd and Dewhurst are the result of extensive experimental studies performed to measure the effect of part symmetry, size, weight, thickness and flexibility on manual handling times (Boothroyd and Dewhurst, 1994). Additional experiments were conducted to quantify the effect of part thickness on the grasping and manipulation of a part using tweezers and the effect of weight on handling time of parts. In terms of insertion, experiments performed by Boothroyd and Dewhurst include the effects of chamfer design, part geometry,

obstructed access and restricted vision on the insertion time of a manual assembly operation.

To successfully extract information form the time standard sheet derived by Boothroyd and Dewhurst, the classification system and codification (explained in Appendix A) employed by the system needs to be understood. The classification system for manual handling is a systematic arrangement of part features in order of increasing handling difficulty levels. The part features that affect the handling times significantly include (Boothroyd and Dewhurst, 1994);

1. Part specifications; part size, thickness, and weight
2. Part attributes; parts that are prone to nesting and tangling, and parts that are fragile, flexible, and sticky.
3. Part external handling necessities; necessity for using two hands, for using grasping tools, for optical magnification, and mechanical assistance.

The classification system for manual insertion and fastening system deals with the interaction of mating parts. Manual insertion and fastening consist of a finite variety of basic assembly tasks such as peg-in-hole, screw, weld and rivet. The design features that significantly affect the manual insertion and fastening times are (Boothroyd and Dewhurst, 1994):

1. Accessibility of assembly location.
2. Ease of operation of assembly tool.
3. Visibility of assembly location.
4. Ease of alignment and positioning during assembly.
5. Depth of insertion.

The time standard data sheet for manual insertion and fastening can be found in Appendix A.

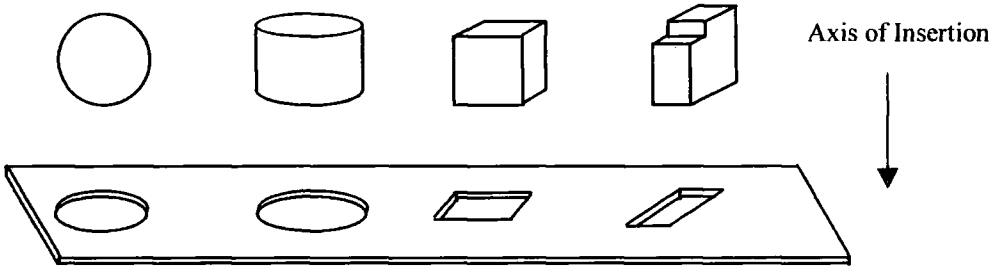
It can be seen from the time standard sheets in Appendix A, that for each two digit code an average handling and insertion time is given, thus setting a time standard to be used to estimate manual assembly times. Distinctive to the work carried out by Boothroyd and Dewhurst is the in-depth investigations into the effects of part features on handling and insertion times.

1. Effects of part symmetry on handling time: One of the principal geometrical design features that affects the time required to grasp and orient a part is its symmetry. Assembly operations describe the mating process of at least two parts; the part being inserted, and the part into which it is inserted. Orientation involves the proper alignment of the part to be inserted relative to the corresponding receptacle. The orientation process can be divided into two groups, alignment with the axis of the part that corresponds to the axis of insertion and rotation of the part about the axis of insertion. As a result Boothroyd and Dewhurst defines two types of symmetry;

   Alpha Symmetry; depends on the angle through which a part must be rotated about an axis perpendicular to the axis of insertion, to repeat its orientation.

   Beta symmetry; depends on the angle through which a part must be rotated about the axis of insertion to repeat its orientation.

   Examples of alpha and beta symmetry are shown in Figure 2-1.



α	0	180	90	360
β	0	0	90	360

Figure 2-1: Alpha and Beta rotational symmetry for various parts

Numerous attempts have been made to define a single parameter that would give a satisfactory relationship between part symmetry and handling time. According to Boothroyd and Dewhurst the simplest and most useful parameter is the summation of alpha and beta symmetries (Boothroyd and Dewhurst, 1994).

2. Effect of part thickness and size on handling time:

   a. Thickness: The thickness of a part is defined as the maximum height of the part with its smallest dimension extending from a flat surface. The thickness of a cylindrical part is generally defined as its diameter, as shown in Figure 2-2. In general if the thickness of a part is greater than 2mm then the part presents no grasping or handling difficulties

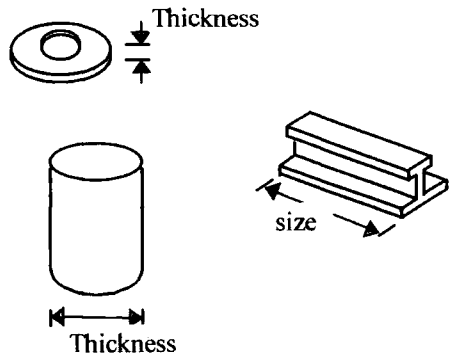


Figure 2-2: Examples of Part Thickness and Size

b. Size (Major Dimension): The size of a part is defined as the largest non-diagonal dimension of the part's outline when projected from a flat surface. This normally refers to the length of the part. Parts are divided into four size categories. Large parts involve little or no variations in time with changes to their size. Medium and small parts display progressively greater sensitivity with respect to part size. Small parts usually result in the use of tweezers. Very small parts involve the use of optical magnification.

   A graphical representation of the effects of part thickness and size on handling time can be found in Appendix B. A numerical expression has been obtained using curve fitting.

3. Effect of Weight on Handling Time: The effect of weight on part handling times has been found to be a time addition to the basic time for grasping and controlling a light part. The effect of weight on part handling can be expressed by the following equation:

$$t_{pw} = 0.0125W + 0.011Wt_h \qquad \textbf{Equation 2-1}$$

Where W (lb.) is the weight of the part and t_h (sec) is the basic time for handling a light part. Average value for t_h is 1.13 sec.

4. Effects of Holding Down: Holding down is required when parts are unstable after insertion or during subsequent operations. It acts as a means of maintaining the orientation of parts, which are already in place prior to or during subsequent operations. Boothroyd and Dewhurst have developed an expression to calculate the basic time needed for an insertion operation where the parts are pre-aligned and self-locating (Boothroyd and Dewhurst, 1994). The time taken to insert a part through two or more stacked parts or assemblies can be expressed as the

sum of the basic time t_b, and t_p, a time penalty. The following equations have been extracted from Boothroyd and Dewhurst data, again by using curve fitting. The graphs can be found in Appendix B.

$$t_b = -0.07\ell nc - 0.1 + 3.7L + 0.75d_g \qquad \textbf{Equation 2-2}$$

Other effects of part features investigated by Boothroyd and Dewhurst that are of particular interest are tool specific. These include the effects of obstructed access and restricted vision on bolting, screwing and riveting operations (Boothroyd and Dewhurst, 1994). Also of interest is the effect of number of threads on time to pick up the tool, engage the screw, tighten the screw and replace the tool. The above information is found in Appendix B.

2.4 Assembly planning systems

Until recently, much effort has been devoted to designing and producing individual parts, whilst relatively little effort has been put towards optimising product assembly. Hence, it is likely that an improvement in this area could yield a great reduction in both manufacturing and assembly cost and time.

Assembly planning can be defined as "an act of preparing detailed instructions for the assembly of a product" (Lin and Chang, 1993). The instructions specify all the details of assembly such as mating features to be used, sequence of part mating operations, fixtures and fixture methods, part placement methods, part alignment methods and part insertion paths. Assembly planning activities in many manufacturing industries are still relatively primitive when compared to advances in other computer aided engineering sectors such as product modelling, product testing (including finite elements, fluid dynamics, and dynamic testing), and machining. The assembly planning process is still very much driven by the experience of process planners. However, the task is becoming increasingly difficult due to the changing nature of the manufacturing environment where the requirement is to produce a large variety of more complicated products with shorter life cycles. The amount of information to be processed is therefore increasing rapidly whilst the time available to process the information becomes shorter.

Cadwell, Ye, and Urzi (1995) analysed the use of computational tools available to support assembly planning in the manufacturing industry. They concluded that many manufacturing systems used by production engineers provided little support to assembly planning. The process of generating an assembly production system from an assembly design via an assembly plan were made manually and communicated between

departments (such as; product design, production engineering, and manufacturing) mostly through conventional channels rather than the electronic channel of information sharing.

However, there are a few systems worth mentioning that offer an integrated approach to assembly planning by linking aspects of product design, DFA, assembly planning and production planning. The Stanford Assembly Analysis Tool (Rommey et al, 1995) system can generate and evaluate geometric assembly sequences of complex products containing from 20 to 40 parts and from 500 to 1500 faces. The Integrated Design and Assembly Planning (Sediel and Bullinger, 1991) system constructs an operation network to reflect the constraints between parts that make up the product. The network is constructed in the form of a graph that represents the relationships among tasks and subtasks for DFA evaluation and assembly process planning. The Concurrent Integrated product Design and Assembly Planning (Zha, Lim and Fok, 1996) system focuses on the integration of product design, manufacturability, assemblability, assembly process planning and simulation with economical and ergonomic analysis and evaluation, the developed system has been applied successfully in lighting products.

The majority of these systems, view assembly planning process in three stages, namely; assembly modelling, assembly sequence generations and assembly sequence evaluation.

Assembly modelling addresses the problem of representing a product in its various assembly states, providing information with regards to geometrical and technological constraints, including mating and precedence conditions. Precedence conditions represent the fact that some components have to be assembled before others otherwise they will interfere with a later assembly operation. It is important to note that precedence constraints are related to operations and not to the parts (Ben-Arieh and Kramer, 1994). Assembly sequence generation deals with the creation of the assembly operations and the linking of assembly operations/processes to components, taking into consideration details of assembly data including mating features used, sequence of part mating operations, fixtures and fixturing, part placement methods, part alignment methods and part insertion paths.

Assembly sequence evaluation is the automatic or manual assessment and ranking of assembly plans taking into consideration external limitations including factory layout and assembly cost. Various models have been used for the generation of assembly plans, including knowledge-based and expert systems, genetic algorithms, simulated

annealing, virtual environments, petri-nets and intelligent design systems, all claiming advantages in different aspects of assembly planning.

The majority of researchers have followed the convention of only examining the first two stages of assembly planning. Other works that consider the final stage of assembly planning as defined above are prone to ignoring the first two stages. This author maintains that both these approaches are flawed as it can be argued that the optimisation of assembly sequences is futile if it is not done in conjunction with resource limitations. Furthermore, the majority of these works do not sufficiently address several issues, including the generation of assembly times and factory limitations.

It is important to say that researchers have divergent ideas on both the ultimate goal of assembly planning systems and assembly plan representation. Some seek to generate only one assembly plan (De Fazio et al, 1990; Baldwin et al, 1991; Wolter, 1990), others several plans (Henrioud, Bonneville, and Bourjault, 1991; De Mello and Sanderson, 1991) and possibly a whole set of assembly plans. Some of these work with sequences (Homem de Mello and Desai, 1990; Baldwin et al, 1991; Wolter, 1991; Zhang and Zhang, 1990) others with trees (Henrioud, Bonneville, and Bourjault, 1991; Park, Kwon, and Chung, 1991), the rest with more general graphs (Miller and Hoffman, 1989). These facts imply many variations in the methods, which change depending on the aim of the research undertaken which varies from the generation of a set of assembly sequences to the generation of optimal assembly plans.

In general, all methods create or use some form of a product database. This is either a free-form database that is restricted to theoretical problems but contains all needed features, or a CAD database, which is industrial but incomplete for running an assembly planning system. Some researchers also use a purely geometric database or a database also containing technological data (Mascle and Figour, 1990). Some researchers also use bill of materials (BOM) or clusters of parts. However, one can say all these methods have a common point; they use a "graph search" method, optionally mixed with an evaluation of assembly plans, resulting in the elimination of assembly plans considered worse than other ones.

2.5 Assembly modelling and sequence generation

This is probably the most researched field in the generation of optimal assembly plans. It addresses the issue of product modelling/representation where a distinction is made between the geometry of components and the description of relations between

components in the final assembly, that is the representation of assembly operations. The process of assembly modelling is a precondition for further analysis of the parts and the generation and evaluation of feasible assembly sequences, and subsequently, assembly plans. There are various methods of representing product models and assembly sequences; essentially they all combine information with regards to the product structure, component features and relations between the components.

There are different approaches to the representation of assemblies, but most researchers use mainly graph-based representation and advanced data structures. In general, the majority of assembly systems designed by various researchers for modelling assembly systems adopt some form of a graphical model. These "graphs" are usually constructed from a more basic information source such as a CAD database, bill of materials (BOM) or simply from information supplied by the user. There are numerous types of graph models; Tree hierarchies (Bourjault and Henrioud, 1992), Diamond representation (De Fazio and Whitney, 1987), AND/OR graphs (Homem de Mello and Sanderson, 1990), nested list (Ben-Arieh and Kramer, 1994), precedence diagrams (Bullinger and Ammer, 1984), assembly constraints graphs (Wolter, 1989) and interference graphs (DeFloriani and Nagy, 1990) are all used to represent feasible assembly sequences.

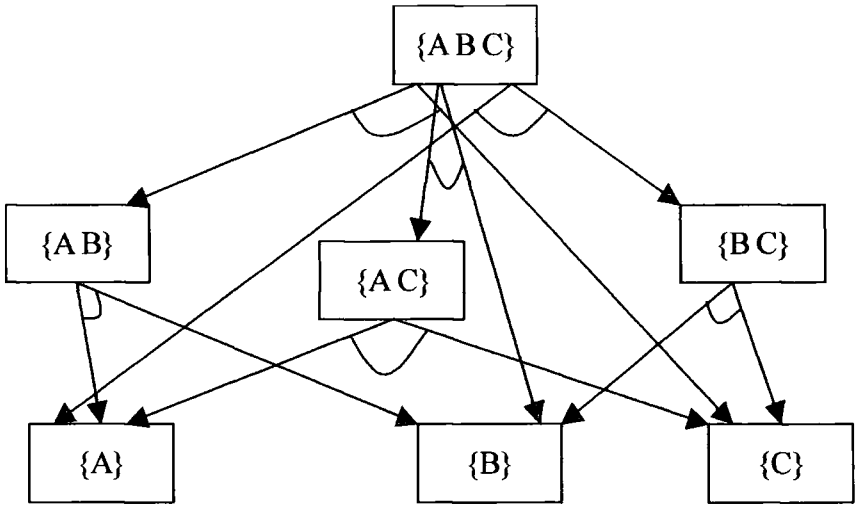


Figure 2-3: The AND/OR graph for a three part assembly (Homem de Mello and Sanderson, 1991)

Approaching the issue of representation in terms of assembly as opposed to disassembly, the procedure proposed by Bourjault obtains all the precedence knowledge about the relations within an assembly by answering a set of structured questions based on his proposed liaison model. De Fazio and Whitney subsequently simplified this process; they successfully reduced the number of questions asked to (2n) against (2^n). Other methods used to represent precedence knowledge of an assembly include set

theory, directed graphs and binary matrix method (Homem de Mello and Sanderson, 1991).

The above-mentioned methods used to represent precedence relationships of an assembly can only represent partial assembly precedence relations. In a bid to generate complete assembly sequences, Homem de Mello and Sanderson proposed the AND/OR graph-representation shown in Figure 2-3 with a disassembly viewpoint. The "nodes" in the AND/OR graph correspond to subassemblies and the "hyperarcs" correspond to assembly tasks in which two subassemblies are joined to yield a larger and more complex subassembly. The hyperarcs point from the node corresponding to the larger subassembly to the nodes corresponding to smaller subassemblies (Homem de Mello and Sanderson, 1991). According to the researchers, the model provides a compact representation for the set of all possible plans, allowing one to traverse the space of all possible assembly plans and therefore have an opportunity to select an optimal sequence and dynamically adapt assembly plans to changing conditions. Gottipolu and Gosh (1995) developed what they termed as an Assembly Sequence Graph (ASG), shown in Figure 2-4, essentially a hybrid of both the liaison graph and the AND/OR graph methods of representation.

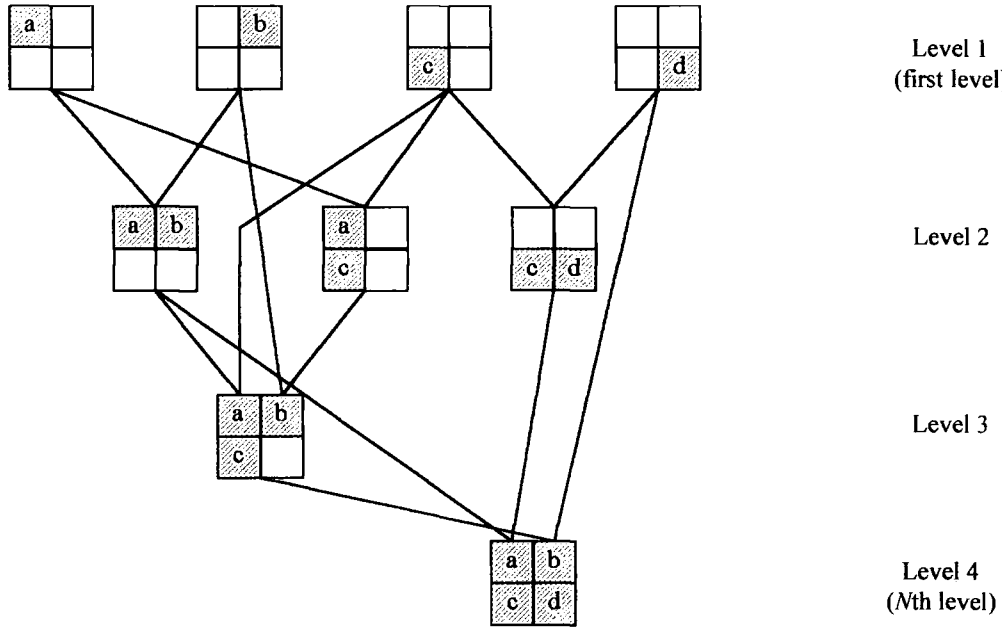


Figure 2-4: Assembly Sequence Graph (Gottipolu and Ghosh, 1995)

The nodes (boxes) in the assembly sequence graph also represent subassemblies. Here the node is made up of a number of *N* cells that correspond to the number of individual parts that make up the assembly. A blank cell implies the corresponding part is not yet assembled and a hatched cell implies the part has been assembled. At the first level

there are N boxes, each with one marked cell, and at the bottom level all cells are hatched, indicating a complete assembly. Although these representations are complete, the AND/OR graph is large and the number of nodes grows exponentially as the part number in the assembly increases (Ben-Arieh and Kramer, 1994).

Other methods worth mentioning include the work of Ben-Arieh and Kramer, who presented a methodology and algorithms for the automatic generation of multiple assembly sequences. The methodology consists of two phases of assembly analysis. In the first phase all feasible sequences of part introduction are generated. These sequences directly correspond to the possible combinations of assembly operations and also define whether the assembly is of internal or external type. Secondly, the sequence from the first phase acts as an input and generates all feasible subassembly combinations. The feasible combinations have to conform to contact, precedence and technological constraints. Lin and Chang (1993) used a three-layer strategy (linking design, planning and production layout) and a complex tree structure to represent the assembly and to generate feasible assembly sequences.

Increasingly, Petri nets are being used for modelling and control of manufacturing systems (Chao and Sanderson, 1995; Thomas, Nissanke and Baker, 1996; Zhang, 1989). There are many Petri net variations such as coloured Petri nets, control nets and object nets. The approach generally includes two algorithms, one to draw and input the Petri net graph of the assembly sequence and the other to apply the liaison matrix approach of the Petri net to obtain the optimal assembly sequence. Zha, Lim, and Fok (1998) use the petri net representation and apply topological, stability and security, partial precedence constraints to generate an optimal assembly sequence. Another method is to generate the Petri net graph based on the assembly AND/OR graph, and apply the straightforward top-down approach to obtain the optimal assembly sequence by linear programming.

More recently Virtual Reality (VR) (Ye, Banerjee, Banerjee and Dech, 1999) has been used for assembly modelling and generation purposes, citing the potential to offer a more natural, powerful, economic and flexible platform than traditional environments to support assembly planning. The advances in the VR environment have been spurred on by the advances in automatically loading CAD data of an assembly design into a VR environment, enabling the rapid prototyping of an assembly and its type. These methods still face a large number of obstacles including accounting for gravity, part weight, irregular geometry and fixtures to name a few. Current studies do not show a significant

advantage over the other methods discussed. However, it is still viewed as a useful tool in supporting assembly sequence generation/planning.

The resultant assembly sequence(s) should be evaluated to identify the most suitable sequence for the given design and assembly operations. Sequence validation generally involves the application of suitable constraints to a set of sequences. Depending on the number and stringency of these constraints, there are often a large number of sequences remaining.

2.6 Assembly sequence evaluation and optimisation

Conventionally, sequence generation constructs the set of all possible assembly steps and then validation criteria are applied to ensure a feasible sequence. The majority of systems that claim to evaluate sequences are generally only validating the sequence for feasibility rather than exploring the merits of different sequences. Others leave the evaluation to the discretion of the user. It is relatively easy to optimise assembly sequences based on a single criterion. It can be argued that sequences should be optimised according to the most prevailing constraining factor based on the economic and environmental situations. However a more suitable viewpoint would to be to use a multi-criteria selection process.

Most of the research done to date deals with the automatic generation of assembly sequences from a CAD system. Only a few researchers have considered the optimisation of assembly sequences. Of this group an even smaller number have taken the issue of multi-criteria optimisation into account, namely; De Mello and Sanderson (1991b), Laperrière and ElMaraghy (1993) and Motavalli and Islam (1997).

To determine the feasibility of an assembly sequence it is necessary to analyse various assembly constraints. These constraints can be broadly classified into two groups: hard and soft constraints (Delchambre, 1996). Hard constraints are those imposed by the topology and geometry of the individual components and the whole assembly. The sequences that satisfy such constraints are deemed feasible assembly sequences. The other group of constraints are usually independent of the geometry of the product and specify additional precedence constraints. The nature of soft constraints is such that they can be used by the designer in decision-making to narrow down the choice to a few good sequences among the feasible sequences generated based on the hard constraints. Such constraints, once numerically expressed, can be used as control parameters for selecting and evaluating assembly sequences from a pool of feasible sequences.

There are two widely used factors for the evaluation and selection of assembly sequences, namely qualitative and quantitative factors. Qualitative factors take into consideration the particular states of an assembly or transitions (assembly operation) in terms of their desirability from a manufacturing standpoint. Qualitative factors include stability of sub-assemblies, reorientation, parallelism, modularity of sub-assemblies, type of assembly and clustering of parts. Quantitative evaluations consider attributes that directly influence the assembly cost and time. Quantitative factors include total assembly time, assembly cost, number of fixtures, number of operators and number of workstations. A detailed description of these factors can be found in 'Computer-Aided Assembly Planning' (Delchambre, 1996).

Barnes, Jared, and Swift (2000) present the development of evaluation criteria based on the identification of a need for metrics that can absolutely determine the suitability of a sequence prior to completion to enable design decisions to be made interactively. The reasoning behind this stems from the dependency of the factors described above on the complexity of the product design. According to the researchers "a complicated design by definition would cost more and take longer to assemble". They cite metrics such as assembly time and assembly cost as useful comparative measures for a set of different assembly sequences for a given design.

There are a few systems of interest that perform an evaluation process. Gottipolu and Gosh approach the issue of evaluation by determining the shortest time or the lowest cost path through the weighted assembly sequence graph. Deleting the unwanted assembly states and tasks or retaining the most desirable assembly states and task then chooses the final assembly sequence. The Zha *et al* approach is based on incorporation and integration with DFA, assemblability analysis, the evaluation approach used in ASPEN (Kanai, Takahashi and Makino, 1996) and the motion time measurement (MTM) pre-determined motion time analysis. The researchers consider quantitative factors by defining cost constraints for the evaluation of sequences based on assembly cost and time. Cost constraints include, the priority index (represents the insertion priority for each part), and the number of workstations.

Schmidt and Jackman (1995) use a multi-echelon simulated annealing procedure for the evaluation of assembly sequences by dividing the problem into two main issues; formulating an objective function for the assembly sequence and developing an optimisation technique for finding the optimal. The derived objective function is based on assembly cost and workstation performance.

Another method that has come to the forefront for assembly planning is the use of genetic algorithms as a search method, giving rise to the possibility of generating a series of optimal assembly sequences. Senin, Groppitti, and Wallace (2000) adopt the AND/OR graph for representation purposes and use genetic algorithms for traversing the graphs generated: their derivation of an objective function for evaluating assembly sequences is based solely on the assembly time. However, the researchers do acknowledge that assembly plans should be ranked according to multiple criteria including resource allocation and line balancing.

The method which is most widely used for optimising assembly planning follows the pattern of representing the assembly sequence by an AND/OR graph or a liaison graph or a hybrid of the two forms of representation Then, heuristic search methods in artificial intelligence (AI) such as the depth-first search, breadth-first search methods, simulated annealing and genetic algorithms are used to obtain the optimal assembly sequence. Recently, the concept of the intelligent design system (Daabub and Abdalla, 1999) (merging a CAD system, expert system and a database management system) has been used to optimise the assembly plans with the aim of providing the design engineer with the vast amounts of knowledge required to design a product that satisfies its functional requirements and can be assembled easily at the conceptual stages of design.

2.7 Assembly systems

The development of the first real example of an assembly line can be attributed to Henry Ford who developed such a line in 1913. But, for 40 years after that, only trial-and-error methods were used for balancing assembly lines. Even by the early 1970s, as revealed in a survey of 95 companies made by Chase (1974), only 5% of them were using published techniques to balance their lines. Since then the situation has not changed much, in a recent article Milas (1990) states that companies design assembly lines manually or by gut-feel or historical precedent. This suggests that either the currently available techniques are inadequate or inflexible to model the actual conditions of assembly lines or the practitioners are unfamiliar with the published algorithms.

The first published analytical statement of the assembly line balancing problem (ALBP) was made by Salveson (1955) and followed by Jackson (1956), Bowman (1960), and Hu (1961). Since then the topic of line balancing has been of great interest to academics.

Today, there are various methods used in industry to accomplish the assembly process, notably Manual single-stations, Manual assembly line, and Automated assembly system. The manual single-stations method consists of a single workplace/station in which the assembly work is performed on the product or some major subassembly of the product; it is generally used in the production of more complex products/components and/or when production is performed in small quantities. Manual assembly lines consist of multiple workstations in which the assembly work is accomplished as the product (or subassembly) is passed from station to station along a line/U-bend. At each workstation one or more workers perform a portion of the total assembly work on the product by adding one or more components to the existing sub-assembly. Automated assembly systems make use of automated methods at the workstations rather than human beings. There are two basic ways in which work is transferred between operator workstations namely non-mechanical lines and moving conveyor lines. For the purpose of this research attention has been given to the Manual assembly line/flow lines, using a moving conveyor/belt to move the subassemblies between workstations at a constant speed.

2.8 Aggregate Assembly Modelling and Planning (AAMP)

Assembly modelling and planning is an important activity within a product development environment. An aggregate assembly modelling and planning (AAMP) method and the corresponding computer based system are being developed at Durham University for the initial design stage of product development (Betteridge, 2000).

This computer-based system comprises aggregate assembly process models, which can function using the overall part and feature geometry definition and uses data concerning assembly resources and standard parts. The system selects and assesses the capability of assembly methods and tools, identifies human resource content and estimates times and costs. The methods and computer-based system provide a link between product design and assembly processes and allow the rapid assessment of various design options and the creation of aggregate assembly plans at an early stage of product development.

This system will then be integrated with the Concurrent Assembly and Process Assessments BLocks for Engineering Manufacture (CAPABLE) (Maropoulos, Bradley and Yao, 1998) a prototype computer-based system that aims to provide planning support in the product development process.

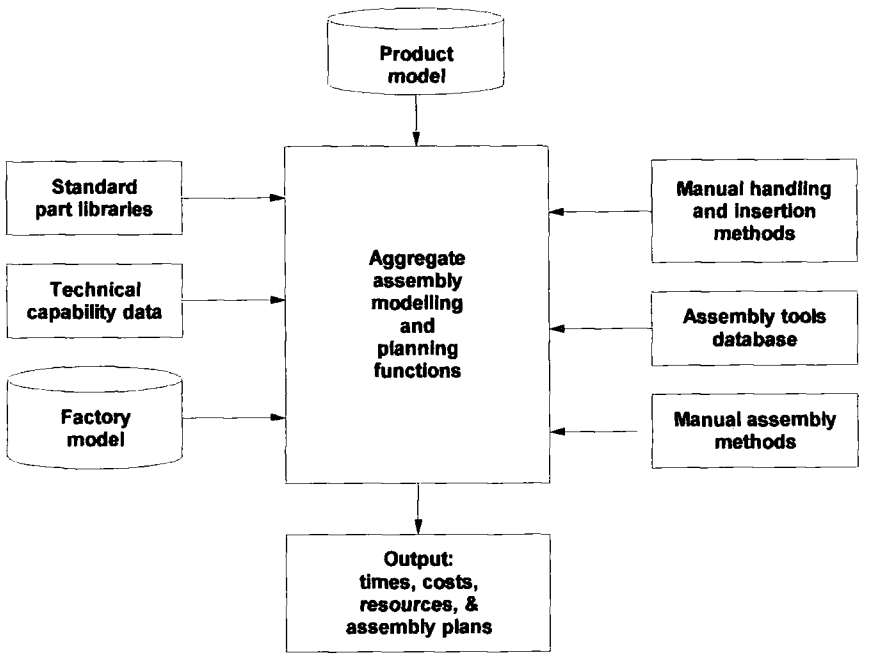


Figure 2-5: Architecture and Functionality of AAMP (Maropoulos and Betteridge 1996)

The Aggregate Assembly Modelling and Planning architecture and functionality is shown in Figure 2-5. Inputs to the system include the product model, assembly process models, manual assembly data, assembly resource models and standard part data. The system provides as outputs assembly times, costs and resources, assembly capability assessment and creates aggregate assembly plans.

The objectives of AAMP (Betteridge, 2000) are;

1. Provide a link between the early stages of product design and assembly processes and methods. Also, the efficient acquisition of accurate assembly planning decisions from limited product information.
2. Derive estimated assembly times, costs and required resources.
3. Create and compare alternative product designs and configurations and also different assembly methods, processes and resources.

Currently the system is capable of;

1. Defining new products, assemblies, components and features. Information is inputted at this stage with regards to parts size, symmetry and handling difficulties.
2. Creating product models, which consist of assemblies, components and features. Alternatively these can be loaded in from a flat file database. New rules are

included for defining features, which include options to input geometric tolerances and capacity to change properties of features.

3. Creating assembly feature relations (AFRs). Options exist to add extra features and redefine assembly feature relations.
4. Deriving types of assembly feature relations. Numerous types of AFRs are supported containing unique information and executing different assembly rules.
5. Deriving information required calculating assembly times, costs and resources dependent on each AFR.
6. Checking the initial geometry and model feasibility of the assembly feature relations.
7. Creating assembly model incorporating the product model. This shows, in a bill-of-material style, assembly order and will contain information with respect to each assembly feature relation. This is defined by attaching an AFR to an assembly node where all features coincide.
8. Definition of "main" and "moving" parts for a connection.
9. Output primary "ideal" handling and insertion assembly times using best in-house tools. The ideal times do not include any considerations regarding assembly area layout or utilisation.

Assembly process models are used to define and simulate the operation of assembly methods such as fastening using screws, riveting or welding. For each assembly operation, it is necessary to model a variety of different aspects, including the cost and time functions, equipment and human resources as well as create precedence rules for the scheduling of the process. A model provides information on how to calculate the time to tighten a bolt, what quality can be achieved in a welded joint and sequences assembly tasks using precedence relationships.

2.8.1 Aggregate Product Modelling

Product data differs in quality of information as well as quantity of detail from conceptual to detailed design. In conceptual design, decisions are made between alternative functional structures, which could meet the specification of the product (Maropoulos and Bradley, 1997). This determines the basic list of components and their principal attributes. It may be neither possible nor desirable at this stage to produce a geometrical representation of the part, since this will depend on factors yet to be considered. At this stage, however, the developer should be able to make some

assessment of the relative manufacturability of alternative conceptual design options in order to select the most appropriate design solution.

The implemented aggregate product model uses an object-oriented representation of the assemblies and components. The individual components are represented using a feature based solid model, which is based on the concept of constructive solid geometry (CSG). In a solid model, a part is built up by the algebraic combination of negative and positive features. Positive features are those, which represent solid material, whilst negative features represent volume where material has been removed, such as holes and slots. The CSG method is more suited to the conceptual design stage, since it lends itself to the gradual addition of more detail through the addition of more negative features to a basic positive feature. In the aggregate product model, the component is defined as a sum of positive and negative features.

A comprehensive list of product features for products can be found in Appendix D.

2.8.2 Assembly Feature Relations and Connections

In terms of aggregate assembly planning, the links between components are based on their mating features. "Assembly feature connection" allows for the specification of features, which connect components together and make a link between these features. Assembly feature connections can be used as a basis for aggregate assembly plans.

Feature relations in general are used to represent all the characteristics of features that may be altered during the development of the design. This includes geometrical information that may be feature specific and connectivity information, which relates to more than one feature.

Within the AAMP, assembly feature relations are referred to as assembly feature connections (AFCs) and those are linked to assembly models. For each connection, an assembly model will contain an appropriate set of assembly methods, which can generate the specific connection.

Moreover, each assembly method requires specific tools and assembly workstation equipment. Each assembly workstation is described by the processes that it can perform and by specific process parameters such as torque rating of a tool or size of part on an assembly fixture. Hence, a link is established between the assembly feature connections of the product model and the assembly resources of the factory.

The assembly feature connections (AFCs) are also used to graphically indicate which component features are linked together using an assembly process, as shown in Figure 2-6.

A comprehensive list of assembly feature connections has been established and methods were developed to automate the creation of AFCs using data from the product features. Current classifications for assembly feature connections as defined by M. Betteridge include placement and insertion assembly operations. Placement operations take into consideration all part handling and orientation, they consist primarily of surface and Plug'n'Target connections. Insertion operations represent the actual assembly operations; these have been subdivided into reversible and permanent assembly connections. Examples of reversible connections include threaded and snap-fit AFCs. Permanent connections include plastic deformation and chemical AFCs. Further information can be obtained from A Methodology for Aggregate Assembly Modelling and Planning (Betteridge, 2000).

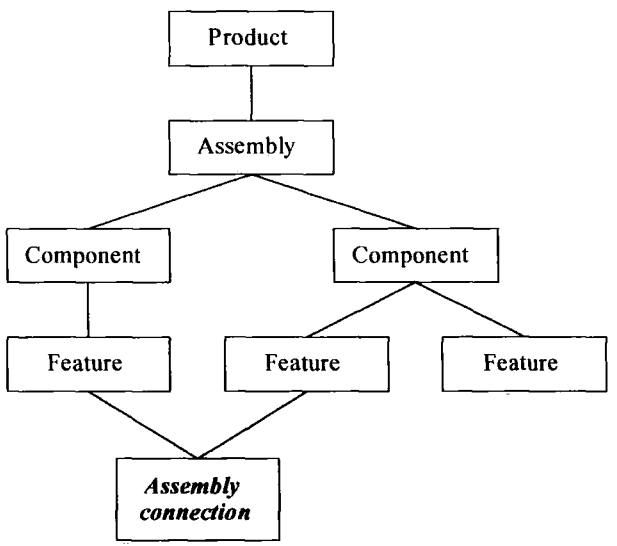


Figure 2-6: Assembly feature connections

2.9 Assembly time generation

There has been a widespread application of the DFA method during the past twenty years and results have shown that assembly time estimations are reasonably accurate for small to medium volume assembly manufacturers (Boothroyd and Fairfield, 1991). However, the method needs to be refined for dealing with large assemblies or high volume production. For example, the time estimated for acquiring, handling and inserting a standard screw is estimated at 9.5 sec using the original DFA method. This time includes acquisition of the screw, placing it in an assembly manually with a couple of turns, acquiring the power tool, operating the tool and replacing it. In a high volume

production situation, it would make more sense for the worker to insert the screw in the power tool and use the tool to insert and drive the screw. This operation would take 7.5 sec. Another example would be if multiple screw insertions were utilised, tool acquisition would only occur once thereby further reducing the assembly time. Variations in tools used to perform assembly operations will also drastically affect assembly times obtained. According to the current Boothroyd and Dewhurst time standards, operation such as screwing, riveting, welding and soldering, have been assigned a single assembly time estimate. Obviously, the assembly times for such operations are wholly dependent on the form of tooling utilised. Also, acquisition and handling of larger products from their storage locations would adversely affect assembly times.

The original DFA method has been extended by allowing for large assemblies (Boothroyd and Fairfield, 1991), medium volume and high volume production and variations with regards to available tooling data, resulting in more accurate estimates of assembly times. However, these developments have been limited due a fear of compromising the effectiveness of the system.

The DFA method compliments other systems currently utilised extensively within industry such as the various forms of work-study applications, that is Predetermined Motion Time Systems (PMTS). Specific advantages include the availability of assembly time expressions with regards to environmental constraints of an assembly workstation (use of fixtures), weight effects on assembly times and the effects of part symmetry. The DFA methodology falls short in terms of its ease of adaptability.

In terms of generating assembly time expressions for particular assembly operations the use of predetermined motion time type systems (MOST SYSTEMS, MTM) provides more structured approach to generating and analysing assembly time expressions for the various assembly sequences of assembly operations. The desirability of PMTS such as MTM and MOST is that they are easy to understand and can be used quickly. PMTS have now become a useful tool for analysing ergonomic conditions, expressing the feasibility of design with regards to manual motions, providing a detailed process description, enabling focused training of work situations, providing a comparison for alternative methods and enabling costing comparisons.

2.9.1 Predetermined Motion Time (PMTS)

Predetermined motion times were initially provided following the introduction of Segurs law (Zandin, 1980). Segurs law states that: "within practical limits, the time required for an average trained worker to perform a basic motion, of specified distance and control characteristics, is constant and is independent of the nature of the work".

Predetermined motion time systems use a procedure which analyses any manual operation or method into the basic motions required to perform it, and assigning to each motion a predetermined time standard. These so-called "basic motions" are dependent on the system being employed.

The major limitation is the use of PMTS systems (in particular, MOST) is their inability to specify factors for weight considerations. According to Dossett (1992), if a person can move a body member with and without a weight in about the same length of time, then the weight factor can be ignored. Also, in most PMTS systems available to date, the effect of part symmetry has not been adequately accounted for. According to Boothroyd and Dewhurst (1996), the comparison of experimental results with MTM shows that the parameters adopted by MTM to define object orientation do not properly account for the symmetry of the part.

The amount of time necessary for assembling a product or component is an important element in cost allocation, production planning and scheduling and evaluation of alternatives. The field of work measurement was developed to establish the time needed for suitably qualified and adequately motivated workers to perform a specified task at a specified level of performance (Genaidy, Agrawal and Mital, 1990). Work measurement techniques encompass time study (direct observation with performance rating), work sampling, standard data, and predetermined motion time systems.

Predetermined motion time systems (PMTS), is a set of time values for the general human motions (Dossett, 1992). The values are typically obtained categorising motions then filming many humans performing these motions. The time is measured and statistically reduced to average times. The coded times are assembled on a data card to be used by design engineers. Data cards comprise coded motions such as reach for an object, grasp the object, move the object, position the object and release the object.

When several PMTS time values are added to compute a total task time individual time errors tend to cancel each other. Accuracy is not a word that can easily be applied to a PMTS since there is no standard against which to compare; there is no standard time to perform human motions. However, MTM-1 is generally accepted as the most

representative of average human motion times in a working environment. In terms of accuracy, almost all PMTS compare themselves to MTM-1. Of the predetermined motion time systems available, the system currently employed by the University of Durham is the MOST system.

2.9.2 Maynard Operations Sequence Technique (MOST)

The MOST work measurement system was developed by Zandin (1980) and consists of three versions namely, Basic, Mini and Maxi (Genaidy, Agrawal and Mital, 1990). For the purpose of this research the basic version has been used.

The BasicMOST consists of three basic sequence models; general move sequence, controlled move sequence, and tool use sequence. In addition to the three basic sequence models, an equipment-handling sequence is available to analyse the movements of heavy objects that require manually operated crane.

1. The general move sequence model consists of three distinct phases as shown in Figure 2-1, which is identified by the following steps:
    a. Reach with one or two hands a distance to the object(s), either directly or in conjunction with body motions.
    b. Gain manual control of the object.
    c. Move the object a distance to the point of placement.
    d. Place the object in a temporary or final position.
    e. Return to final location.

Human Motions	Code
Get	ABG
Put	ABP
Return	A

Table 2-1: General Move Sequence

2. Controlled Move Sequence describes the manual displacement of an object over a controlled path. MOST defines a "controlled path" by the movement of an object restricted in at least one direction by contact with or an attachment to another object. The controlled move sequence, like the general move has three phases, as shown in Figure 2-2:

Human Motions	Code
Get	ABG
Put	MXI
Return	A

Table 2-2: Controlled Move Sequence

3. The Tool Use Sequence Model is a combination of the general move and controlled move models describing the actions performed with tools. The tool use sequence model follows a fixed sequence of sub-activities codified in five main phases, as shown in Figure 2-3.

Human Motions	Code
Get object or tool	ABG
Place object or tool	ABP
Use tool	User defined
Aside object or tool	ABP
Return	A

Table 2-3: Tool Use Model

The tools covered by MOST system data cards are listed on Table 2-4 shown below. Although the tools listed cover a variety of applications there are obviously a greater number of tools in use in areas where MOST is applied. However, this problem is easily overcome by associating index numbers of tools, which are operated using similar body motions. Otherwise, the index values can be customised through a process of Index Value Development (MOST SYSTEMS User Manual, 1990).

Parameters	Tools
Fasten./Loosen	fingers, screwdrivers, wrenches, Allen key, ratchet, hammer and power wrenches
Cut	knife, scissors, and pliers
Surface Treat	cloth/rag, brush, and air hose
Measure	Profile gauge, fixed scale, veneer callipers, feeler gauge, steel tape, and micrometers
Record	pencil, pen, and chalk
Think	mental process

Table 2-4: MOST: Tools covered

The details of the three sequence models are described on Table 2-5. The sequence models, in addition to describing the motions employed, provide the total time value for the activities by using index numbers. An index is placed after each sub-activity in the sequence and represents the time allowed for the sub-activity. The tables provided on Basic MOST (including mini and maxi) systems data cards serve as a reference for identifying the appropriate index values.

Parameter	Symbol	Description
Action Distance	A	Covers all spatial movement or actions of the fingers, hand, and feet, either loaded or unloaded. Any control of these actions by the surroundings requires the use of other parameters.
Body Motion	B	Refers to either vertical motions of the body or the actions necessary to overcame an obstruction or impairment to the body movement
Gain Control	G	Covers all manual motions (mainly finger, hand, and foot) employed to obtain complete manual control of an object and to sequentially relinquish that control. The G parameter can include one or several short-move motions whose objective is to gain full control of the object(s) before it is to be moved to another location.
Place	P	Refers to actions at the final stage of an object's displacement to align, orient, and/or engage the object with another object(s) before the control of the object is relinquished.
Move Controlled	M	Covers all manually guided movements or actions of an object over a controlled path.
Process Time	X	Occurs at that portion of work controlled by processes or machines and not by manual machines.
Align	I	Refers to manual actions following the controlled move or at the conclusion of process time to achieve the alignment of objects.
Fasten	F	Refers to mechanically assembling one object to another using finger, hand, or a hand tool.
Loosen	L	Refers to mechanically disassembling one object to another using finger, hand, or a hand tool.
Cut	C	Describes the manual actions employed separate, divide or remove part of an object using a sharp-ended tool.
Surface Treat	S	Covers the activities aimed at removing unwanted material or particles from, or by applying a substance, coating, or finish to, the surface of an object.
Measure	K	Refers to the actions employed in determining a certain physical characteristic of an object by comparison with standard measuring device.
Record	R	Covers the manual actions performed with a pencil, pen, chalk or any other marking device for the purpose of recording information.
Think	T	Refers to the eye actions and the mental activity employed to obtain information (read) or to inspect an object.

Table 2-5: MOST Systems description

This process is referred to as parameter indexing. Adding together the index numbers, and multiplying the sum by ten obtain the time value for each sequence. This yields the time in TMU, the unit of time in MTM systems.

The manual application of MOST consist of the following steps:

1. Observe and document the workplace and method of application.
2. Select the sequence model.
3. Identify index values from MOST data cards.

4. Obtain the time value by adding the index number and multiplying the sum by 10.

The computerised system requires the analyst to gather work place information and key the information into the computer database. The activity being measured is documented in text format, starting with a key word that designates to the computer the sequence model to be used as well as the values for selected sequence parameters. Keywords and method description conforms to Basic English sentence structure and engineering technology. Utilising the work area data, the computer calculates the standard time for the activity being studied. The Basic MOST computer system is 2-5 times faster than the manual application.

2.9.3 MOST System Calculations: An Example

An assembly worker gets a handful of bolts from a parts bin located within reach and places a bolt through each of the six holes. The assembler puts the rest of the bolts back in the bin. He/she then reaches a handful of washers from a parts bin located within reach and places one on each of the six bolts located four inches apart. He/she puts the rest of the washers' back into the bin.

Task Description	MOST sequence	Parameter index summation	Multiply by No. of parts	Convert to TMU values (10)
Get a handful of bolts form a part bin	$A_1B_0G_3$	4 (1+3)		40
Insert through ***six*** holes located 4" apart	$A_1B_0P_3$	4 (1+3)	24 (x6)	240
Return remaining bolts into the part bin	A_0	0		0
Get a handful of washers form a part bin	$A_1B_0G_3$	4 (1+3)		40
Place on ***six*** bolts located 4" apart	$A_1B_0P_1$	2 (1+1)	12 (x6)	120
Return remaining bolts into the part bin	A_0	0		0
	Summation of TMU values	**440**	**Time in seconds**	**16.1**
If entire sequence occurs twice	**Summation of TMU values**	**880**	**Time in seconds**	**32.2**

Table 2-6: Assembly task using MOST

The time value for the activity described is calculated by adding the time related index numbers and multiplying by 10, as shown in 2-6. To obtain is seconds, the TMU (Time Measured Units) values are simply divided by 27.3.

2.10 Conclusion

This thesis adopts the definition of assembly planning provided by Delchambre, "Assembly planning outlines the nature and the succession of the operations necessary

to assemble a product", but further extends the definition of assembly planning to include to assembly resource planning. It is the view of this author that:

- for any assembly planning system to be tangible within industry
- for the system to be functional at the earlier stages of design
- for the selection of a product design via its assembly plans

It needs to be capable of generating assembly plans that mimic economic and factory constraints depending on the assembly system practiced. Here, assembly system refers to the methods used in industry to accomplish assembly processes such as manual single-station assembly, manual assembly lines and automated assembly systems. This thesis presents an assembly planning system; CAPABLE*Assembly*.

3 CAPABLE*Assembly*: System overview

3.1 Introduction

Despite a considerable number of existing assembly planning systems; Standard Assembly Analysis Tool (Rommey et al, 1995), Integrated Design and Assembly Planning (Sediel and Bullinger, 1991) and Concurrent Integrated Product Design and Assembly planning (Zha, Lim, and Fok, 1996), process engineers in industry today still outline and route assembly sequences manually. The majority of these systems, if used at all, are only brought in to play at the detailed stage of design, immediately prior to production, thus defeating the objective of such computer aided engineering tools. Although it is obvious that process designers/engineers would appreciate a dedicated, computerised tool to speed up the design/redesign of new products (whilst decreasing assembly time and related process cost), through the representation and handling of assembly sequences, one has to keep in mind, the majority of companies have a large extent of in house knowledge with respect to handling and sequencing of parts. As a result, most companies would only be interested in systems that are easily adaptable to processes within the company. Also, although most firms today have Computer-Aided Design (CAD) facilities, the majority of conceptual design is still performed manually through a series of brainstorming meetings. Indeed, CAD models of products are predominantly created at the embodiment design stage of design.

However, for a great deal of redesign work CAD models are readily available. It is fair to say the lack of use of assembly planning aids is the result of inconsistency and complexity of current planning methodologies. The challenge therefore to researchers, lies in the development of flexible systems, and/or methodologies, for the conceptual stages of design capable, of performing robust and comprehensive assembly planning analysis, whilst maintaining an inherent simplicity. CAPABLE*Assembly* is a prototype of such a system. This chapter presents an overview of CAPABLE*Assembly*.

CAPABLE*Assembly* is essentially targeted at two groups of personnel within industry, namely the product design engineer, and the assembly process planner. Such a tool is desirable to the product design engineer because it provides a useful means of trading-off alternative designs in terms of assembly time and thus cost. Also, the assembly process used to build a part is directly linked to the part's features. For example, if a part were to be assembled using screws, the mould used to create the shells of the part

would be different to say, if the part was to be welded together. Thus, such decisions will have to be made by the product design engineer.

To an assembly process planner the appeal of CAPABLE*Assembly* is increased. The system aims to provide the assembly processes planner with host of feasible assembly process plans that best suits his/her available resources. It also provides a means of providing information with regards to effects of altering available resources without physically having to change the factory layout. For example, a brown field analysis (see Section 6.6.2) will give the planner a series of optimal assembly process plans based on the current/user-specified factory/shop floor layout. However, a green field analysis (see Section 6.6.2) will provide the user with a pool of assembly process plans based on unlimited available resources.

When designing/developing an assembly planning system (or any software), in order to ensure its consistency and completeness, it is important to derive an ideal architecture of modules, to act as a comprehensive and automated centre of activities, capable of gathering relevant information of all aspects of production.

3.2 System requirements

CAPABLE*Assembly* is an assembly-planning tool used to provide the design/product engineer with a series of optimised assembly plans at the conceptual stages of design. The system addresses the issue of assembly product modelling and development, formation and evaluation of assembly processes including assembly line balancing and factory layout. CAPABLE*Assembly* describes a concurrent assembly planning system where assembly plans are generated simultaneously as the product is being developed. Product designers detail the geometry of the components and their relationships within the assembly model, whilst simultaneously receiving feedback on plans originated from the current product configuration. Each change in the product model potentially affects the set of assembly plans that can be generated.

The majority of assembly planning systems use some form of optimisation algorithm based on various mathematical models as described in Chapter 2 to systematically or randomly search and analyse a predefined solution space or one initiated at runtime. The main problem experienced by currently available assembly planning systems include:

1. Exploring a predefined or generated search space that may or may not contain possible/feasible/optimal assembly plans.

2. Minimising computational time whilst maintaining a reasonable degree of accuracy in results generated, independent of the number of constituting components for a given product.

The issue of exploring the solution space generated for a given product model is further exemplified when assembly plans are generated for products with a large number of components with a correspondingly large solution space. The most common compromise adopted by many systems has been to increase the number of assumptions made, thus simplifying the problem definition and decreasing the search space. Other methods limit the number of mathematical objectives used to explore the search space, thus reducing the computational time by streamlining the representation of the problem solution.

Fundamental to the ideology behind CAPABLE*Assembly* is the use of a twofold optimisation process to handle the solution space; assembly sequence and line balancing optimisation methods to generate optimised assembly plans. Here the generation of optimal assembly plans has been split into two; the optimisation of assembly operations and the loading of assembly operations to workstations. The optimisation of assembly sequences is used to initially decrease the search space. However, the generation of optimised assembly sequences does not in itself define an optimised assembly plan. It provides a useful starting point and the best means of decreasing the size and regularising the pattern of the solution space.

It is this assembly sequence that is used to generate the preceding solutions space used for the generation of assembly plans. Here, due to the decreased solution space and depending on the heuristic or exact methods used for optimisation, a larger number of mathematical model objectives can be used to generate an optimised assembly plan. Furthermore, although less important academically but justifiably more prominent in industry, due to the optimisation algorithm adopted by CAPABLE*Assembly*, the system is capable of providing a number of "good" assembly plans from a pool of feasible assembly plans.

The planning of CAPABLE*Assembly* is governed by the research objectives outlined in Chapter 1. The design specifications for CAPABLE*Assembly* are that the system must:

1. cater for product assembly modelling to illustrate the product modelling concept in the assembly domain complete with facilities to respond easily to changes in design if the system is to be used in the early stages of design.

2. fully emulate assembly operations as performed on the shop floor.
3. generate optimised assembly sequences from a pool of feasible assembly sequences
4. optimise the loading of assembly systems by analysing the possible option sequences and factor limitations.
5. provide accurate assembly times with detailed process analysis.

3.3 System structure

The workings of CAPABLE*Assembly* can fully be described under three broad headings, (*i*) analysis tools for assembly, (*ii*) assembly product modelling and product data storage, and (*iii*) automated assembly process planning. Figure 3-1 outlines the overall structure of CAPABLE*Assembly*.

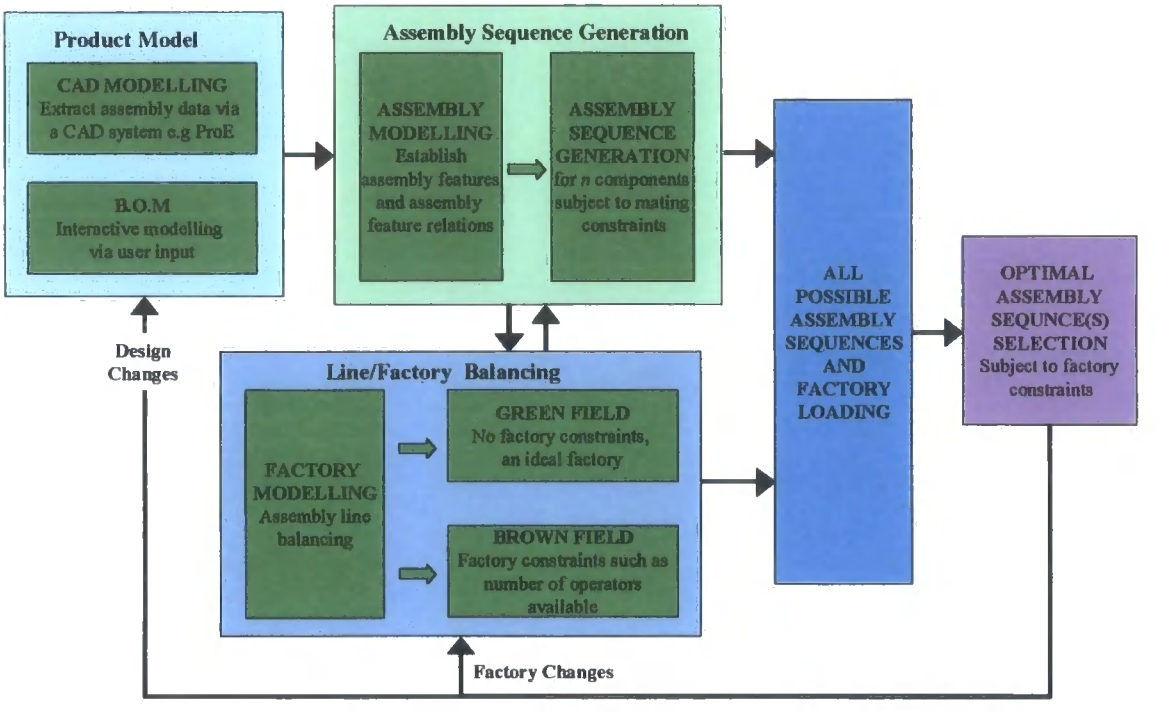


Figure 3-1: Structure of CAPABLE*Assembly*

The product model is created and stored in a flat file database format using the NEXPERT Object shell, which runs on a Unix/IRIX platform. The subsequent assembly and resource planning analysis is performed using Visual C++, which uses Windows2000. The operating system was transferred from the Unix platform to the Microsoft windows platform to make the system more accessible, user friendly, and portable. The switch to C++ arose when a greater degree of control over computational methods and structures was required.

3.3.1 Analysis tools for Assembly

In traditional systems the heart of a program consists of the algorithm and data. The execution of the program is done in a concise manner laid down by the rules of the algorithm. In comparison the NEXPERT Object provides an intuitive means of representing product models by the use of objects and classes. However, these objects need to have some form of reasoning to govern the generation of objects and object hierarchy. A database containing explicit knowledge of assembly processes in a specialized domain, and a reasoning or interface engine which can access the database, is used to govern the generation of objects.

CAPABLE*Assembly* uses Smart Elements, a software system that contains NEXPERT Objects, with a user-interface building tool, Open Editor, and a knowledge database, to facilitate the assembly product modelling process. Knowledge with regards to the inherent characteristics of assemblies, components, resources and AFCs, are stored in the knowledge base. Whilst, the use of a reasoning engine proved to be more than adequate for assembly product and process modelling purposes, it is however inadequate for high level optimisation problems encountered when solving intractable problems such as assembly line balancing and sequencing. As a result a different system/language (Visual C++), which provides greater flexibility to the programmer and is also capable of handling object-oriented programming with facilities for the design of a user interface, was chosen for the generation and optimisation of assembly process plans.

The aggregate product model uses an object-oriented platform for the representation and management of assemblies, components and assembly operations where objects and classes are used to describe assembly entities. Properties are used to describe the characteristics of objects and classes, storing relevant information about specific objects and classes such as assembly features and dimensions. The system creates objects dynamically during the modelling process, allowing the creation of assembly structures that are not yet known, by creating dynamic links between objects, classes and relevant databases to reflect the changing relationships during the assembly modelling process. Generic rules are used to govern and manipulate the behaviour of the objects and classes and their associated methods.

The simplest way to tackle assembly planning is to view it in tandem with the optimisation of assembly sequences, and loading of assembly operations on an assembly line, with or without a pre-defined factory layout. However, both these

problems are intractable and as such an exact solution cannot be found using exhaustive search methods (which simply visit all points in a search space and retain the best solution visited). Heuristic search methods in artificial intelligence (AI) such as the depth-first search, breadth-first search methods, simulated annealing and genetic algorithms can be used to obtain the optimal assembly sequence. Two forms of general-purpose heuristics have been used to obtain an optimised assembly sequence and assembly line respectively. The planning algorithms applied, simulated annealing and genetic algorithms, provide fast recalculation times with reasonably accurate results, providing product designers with real-time feedback about process performance.

3.3.2 CAD: assembly modelling and database

3.3.2.1 The product assembly modeller

In conceptual design, decisions are made between alternative structures, which could meet the functional specification of the product (Bradley and Maropoulos, 1995). This determines the basic list of components and their principal attributes. It may be neither possible nor desirable at this stage to produce a geometrical representation of the part, since this will depend on factors yet to be considered. At this stage, however, the developer should be able to make an assessment of the relative manufacturability of alternative conceptual design options in order to select the most appropriate design and process solutions. This is important since a large proportion of life-cycle cost is determined during conceptual design.

Assembly modelling is a procedure for describing the assembled state of a given product model/assembly in terms of its basic assembly connections and features. An aggregate assembly modelling and planning (AAMP) method and the corresponding computer based system has been developed at Durham University under the supervision of Prof. P.G. Maropoulos by Michael Betteridge (Betteridge, 2000) for the initial design stage of product development. The starting point for the development of an aggregate product model is obtained from a bill of materials (interactive modelling via user interface) or a CAD product model. The aggregate product model is created by extracting assembly features from the constituting components of the product. Assembly feature connections (AFCs) are subsequently established between the components assembly features. These AFCs emulate assembly processes or operations performed on the shop floor of manufacturing firms.

3.3.2.2 Database organisation

Conventionally, the database for CAD systems contains basic graphics elements such as points, lines and curves. For assembly modelling purposes such information is stored within the expert system. The databases used in CAPABLE*Assembly* mainly carry AFC related information and resource model data. This includes information such as handling and insertion times based on handling and re-orientation difficulties for all AFCs considered and the topological position of available resources within a work cell/workstation.

The handling databases contain basic assembly times and time penalties for handling parts based on parameters such as the size and weight. The insertion databases store assembly times for performing actual joining operations, such as welding and threaded AFC. The times stored for each AFC is tool based, for example the operation time for a riveting operation, will vary depending on whether a riveting gun or manual lever riveter is used. The database is also used to store product models created at run-time if desired.

3.3.2.3 User interface

The user interface used for the development of the aggregate product model is as designed by Michael Betteridge for the existing AAMP system. The user interface is designed using Smart Elements; based on NEXPERT OBJECT, which supports a range of representation features. NEXPERT provides the *tutor interface* to access the knowledge base and interference engine used to design the user interface. The domain is modelled in terms of objects and classes and slots (better known as properties). Slots are used to store all the information NEXPERT gathers from the user. The main window for the AAMP system and detailed information can be found in Betteridge, 2000).

3.3.3 Automated Assembly process planning

CAPABLE*Assembly* is an assembly planning system used for the generation of optimal assembly plan(s). A connectivity model derived from the product model is used to represent the assembly sequences. Heuristic search methods namely simulated annealing and genetic algorithms are then used to find optimal solution(s). The system aims at providing a robust DFA tool to aid the design or redesign of a product during the initial stages of design. It takes into consideration all the advantages and disadvantages discussed in the previous sections. Furthermore, the system supplies a separate module for the use of standard parts and the optimisation of standard part

handling and insertion operations. A sophisticated method for the estimation of assembly time is inherent to the generation of assembly operations.

3.3.3.1 Resource planning

Resource planning is concerned with the selection of production means adequate for performing all the assembly operations specified by the assembly planning (in a sequence compatible with the assembly plan it supplies) on parts specified by the design for assembly, while meeting the production volumes set by marketing. A production workshop can be set up using various topologies such as a combination of assembly lines and work-cells, and stand-alones. CAPABLE*Assembly* uses a resource planner to generate a production workshop layout based on information provided by the design engineer or stored within its database. The topology of the line connecting various workstations is seen as a natural topological expression of the assembly process. Hence, the resource planner provided within CAPABLE*Assembly* is used to create the objects that fully describe an assembly line for a single product. These include operators, tools, transportation modes (typically a conveyor belt), and workstations. Each of these objects stores characteristic properties that define the scope of their use during runtime. For example, each workstation has a workstation cycle time, which determines the output rate of the assembly line.

3.3.3.2 Sequencing of assembly operations

The main purpose of the research described herein is to create a system suitable for the automatic generation of an optimal assembly sequence for a given product at the early stages of design. The method is based on the creation of an aggregate product model and the subsequent extraction of contact, precedence and technological relationships from the aggregate product model to create a connectivity model. The extraction of such relationships facilitates the generation of an initial rudimentary assembly plan, which reduces the search space. The generated assembly plan is then refined through a series of optimisation methods using simulated annealing. The simulated annealing algorithm seeks to optimise an assembly rating variable, which includes functions for reorientation, parallelism and stability.

The ability of the system developed to quickly generate and optimise assembly plans locally as well as globally, when various criteria are enabled or their relative importance is changed, makes it an effective tool for simultaneously considering several manufacturing considerations at the design stage. As can be seen from Figure 3-1, the

assembly sequence generation module provides the input for the assembly line balancing module.

3.3.3.3 Assembly line balancing

The system described herein uses genetic algorithms to generate optimised assembly plans. It employs the minimisation of cycle time and number of workstations as the basis of for the initialisation of the genetic algorithm and aims to find the best solutions that lead to the maximum production rate and minimum workstation workload variance, with maximum work-relatedness. The distinction between this methodology and other techniques lies in the way the problem has been designed. It does not seek to simply produce assembly plans based on the minimisation of cycle time or number of workstations; rather it merely uses these parameters for the generation of the initial population, the main interest being in the 'goodness' of the solutions/assembly plans generated. The input to the system is an optimised assembly sequence, providing the designer with a good visual idea as to how the assembly operations would be loaded using a green field site, thus reducing the search space and computational time. The system runs in two modes, green and brown field, as shown in Figure 3-1. If a green field is employed the workstations are devoid of restrictions from operator skill, tooling requirements and shop floor space. If a brown field is employed the loading of assembly operations to workstations is limited to tool availability and operator skill.

3.3.3.4 User Interface

The user interface for the optimisation modules take two formats; an executable file generated within Visual Studio or using MS-DOS. When running in DOS the format is quite simple; the user is presented with a menu, and depending on the choice, the user is taken to the appropriate section of the modules.

The executable file is in essence a dialog-based project created using Visual C++. The user is guided through a series of dialog boxes where he/she is prompted when information from the user is required. As with the NEXPERT system the domain is modelled in classes, objects, and properties. Visual C++ access to time and factory databases is maintained via Microsoft Access. The user interface is largely elementary and has only been developed to the extent where a user can perform the analysis in what is hoped an intuitive manner.

3.4 Conclusion

In this thesis, an intelligent assembly planning system is used for the generation of optimal assembly plan(s) is presented. A connectivity model derived from the product model is used to represent the assembly sequences. Heuristic search methods namely simulated annealing and genetic algorithms are then used to find optimal solution(s). The system aims at providing a robust DFA tool to aid the design or redesign of a product during the initial stages of design. It takes into consideration all the advantages and disadvantages discussed in the previous sections. Furthermore, the system supplies a separate module for the use of standard parts and the optimisation of standard part handling and insertion operations. A sophisticated method for the estimation of assembly time is inherent to the generation of assembly operations.

4 Aggregate Assembly Modelling and Representation

4.1 Introduction

This chapter presents the building blocks of CAPABLE*Assembly*; the aggregate product model, the connectivity model and the assembly time generation algorithm. The work presented herein addresses both the complexity and incompleteness of product data/models and resource information. This is a typical problem faced by process engineer/designers, when drafting an assembly process plan at the earlier stages of design.

As industry increasingly moves away from exclusive geometric modelling practices to more process oriented modelling (a sort of feature-based modelling), the problem we are faced with is how to economically develop a pseudo-product model that captures all process features and relationships, whilst keeping the generic nature of the product at the early stages of design. The aggregate product model can be viewed as the first attempt at the derivation of such a product model. The reasoning behind the shift towards product process modelling (Ling et al, 1999) is embodied in the ideology of a feature-based aggregate product model.

This chapter describes the mapping of a product's assembly features onto generic assembly processes to form a "connectivity model" of the product. The connectivity model is used to generate feasible assembly sequences, taking into consideration previous assembly knowledge, precedence relationships, contact relationships and technological constraints for multiple mating conditions.

The layout of this chapter is as follows:

- Section 4.2 introduces the concept of aggregate product modelling for assembly. It details the assembly modelling and representation method used to define product assembly features and assembly operations.
- Section 4.3 outlines the derivation of the connectivity model, which is a relational product model, derived from the aggregate product model for generating feasible assembly sequences.
- Section 4.4 details the assembly time generation algorithm (AGA), used within both the aggregate product model and the connectivity model to estimate assembly operation times.

- Sections 4.5 and 4.6 are subsections of Section 4.4, but as they introduce the concept of standard parts and standard part assembly methodology respectively, a section has been assigned to each concept in its own right. They explain the methods used to generate standard assembly operations, and standard assembly times used to estimate assembly operation times (employed by AGA).
- Finally, Section 4.7 pulls together all the conclusions drawn from this Chapter.

4.2 Aggregate product modelling for assembly

A product model, as vital as it is, is just one of the information resources needed to fully realize the integrated operation of a manufacturing company. In many manufacturing companies, it is not uncommon to find design and process engineers working concurrently on different aspects of a given product in different countries and/or continents. This has driven the need to transfer product model data quickly, efficiently and accurately through a variety of media, making the minimisation of data stored within product models essential. To a design engineer considering the assemblability of a given product, the need here is for a product model that contains product data associated with the assembly process. This is done by modelling a product in terms of its assembly features.

In conceptual design, decisions are made between alternative structures, which could meet the functional specification of the product (Bradley and Maropoulos, 1995). This determines the basic list of components and their principal attributes. It may be neither possible nor desirable at this stage to produce a geometrical representation of the part, since this will depend on factors yet to be considered. At this stage, however, the developer should be able to make an assessment of the relative manufacturability/assemblability of alternative conceptual design options in order to select the most appropriate design and process solutions. This is important, since a large proportion of life-cycle cost is determined during conceptual design.

An aggregate assembly modelling and planning (AAMP) method and the corresponding computer-based system has been developed at Durham University under the supervision of Prof. P.G. Maropoulos for the initial design stage of product development (Betteridge, 2000). The AAMP system is used as the basis for product modelling activities within CAPABLE*Assembly*. The starting point for the development of an aggregate product model is obtained from a bill of materials (interactive modelling via a user interface) or a CAD product model. In general, an aggregate product model is created by extracting or adding assembly features from the constituent components of

the product. Assembly feature connections (AFCs) or mating relationships are subsequently established between the components' assembly features. The hierarchical structure of the product model is maintained in the aggregate product model. The AFCs emulate assembly processes or operations performed on the shop floor of manufacturing firms. The steps involved in the generation of an aggregate product model are shown in Figure 4-1.

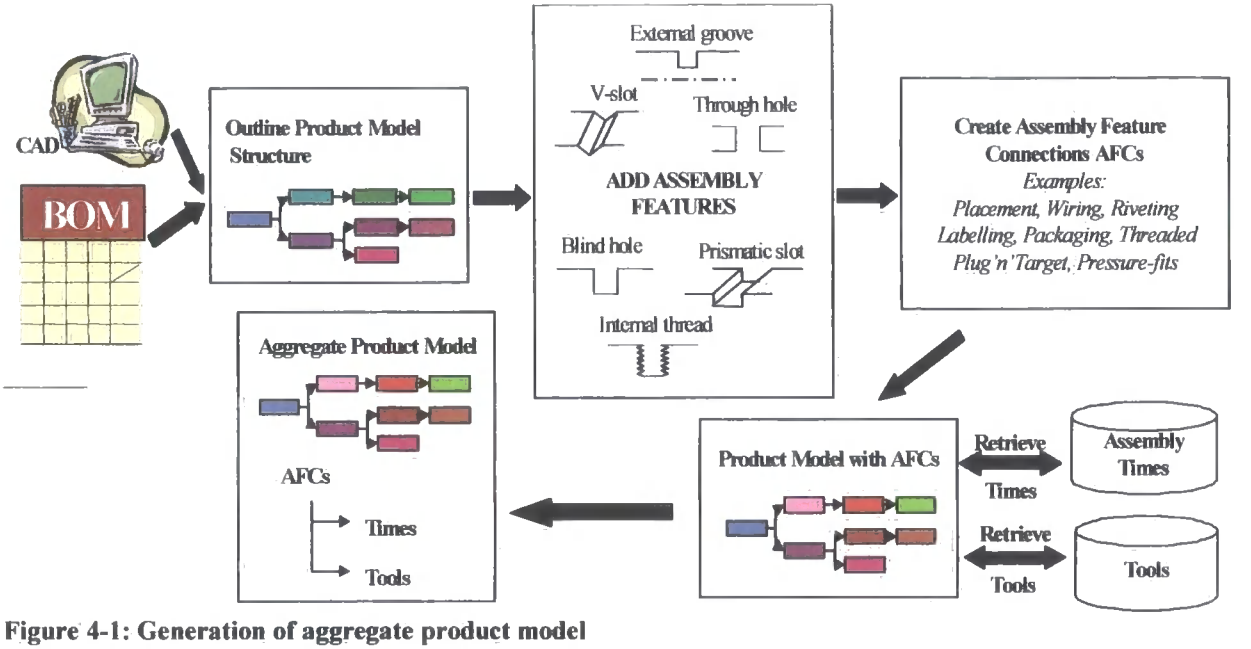


Figure 4-1: Generation of aggregate product model

4.2.1 Assembly modelling and representation

Assembly modelling is a procedure for describing the assembled state of a given product model/assembly in terms of its basic assembly connections and features. The majority of products produced by today's manufacturing firms can be considered to have a multi-level product structure. In such a structure, final products are composed of subassemblies and components, and each subassembly is in turn made of subassemblies and/or components.

The aggregate product model uses an object-oriented method for the representation and management of assemblies, components and assembly operations, where objects and classes are used to describe assembly entities. Properties are used to describe the characteristics of objects and classes, storing relevant information about specific objects and classes such as assembly features and dimensions. The system creates objects dynamically during the modelling process, allowing the creation of assembly structures that are not yet known by creating dynamic links between objects, classes and relevant databases to reflect the changing relationships during the assembly modelling process.

Generic rules are used to govern and manipulate the behaviour of the objects and classes and their associated methods.

There are three major types of geometric CAD models (Mäntylä and Shah, 1995) namely;

1. Graphical models, to support generation of engineering drawings. This group includes wire-frame models. As models of 3D solids, graphical models are poor as it is possible to make models that are valid and yet meaningless because they have several interpretations as a solid body.
2. Surface models, to support the design and manufacture of complex surfaces, used mostly for machining purposes.
3. Solid models, to capture completely the three-dimensional geometry of a solid physical object in order to support higher levels of functionality and information.

In CAPABLE*Assembly*, the individual components are represented using a feature-based solid model. There are two main types of solid models of interest currently employed within industry, boundary representation (B-rep) and constructive solid geometry (CSG). For the purpose of this research, a solid model is based on the concept of CSG (Mäntylä and Shah, 1995).

In the feature based solid model (Bradley and Maropoulos, 1997; Yao, Bradley and Maropoulos, 1998) the part is built up by the algebraic combination of negative and positive features. Positive features are those that represent solid material, whilst negative features represent volume where material has been removed, such as holes and slots. The CSG method is more suited to the conceptual design stage, since it lends itself to the gradual addition of more detail through the addition of more negative features to a basic positive feature.

In the aggregate product model, the component is defined as a sum of positive and negative features. Examples of positive features include: cylinder, prism, moulded and sheet. Negative features are regularly updated, examples include: blind holes, external and internal threads, external steps, through holes, external groves and V-slots. A list of currently available assembly features can be found in Appendix D. A diagram showing the data and hierarchal structure of an aggregate product model generated in CAPABLE*Assembly* is shown in Figure 4-2.

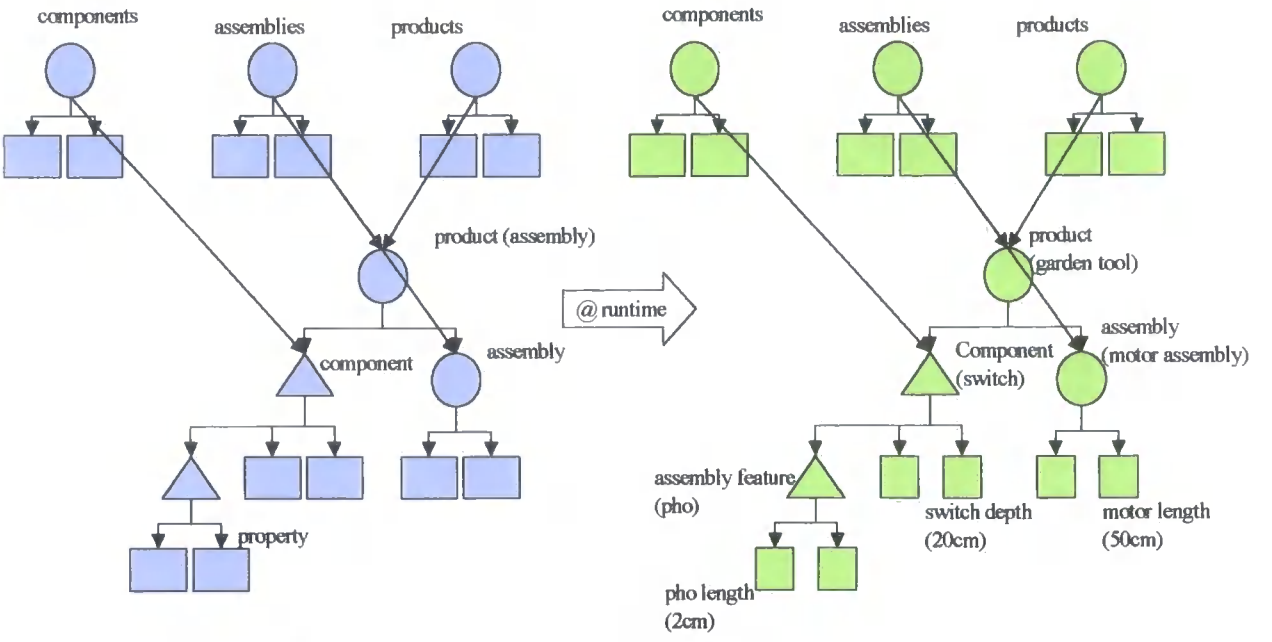


Figure 4-2: Assembly modelling and representation in CAPABLE*Assembly*

The circles represent classes and subclasses, which can be viewed simply as a grouping or generalisation of a set of objects. There are four main classes, three of which are shown below, the fourth being that relating to assembly feature connections (covered in the following section). The triangles represent objects and sub-objects; they are instantiations of classes or specific members. The rectangles are the properties of the objects and the squares can be viewed as values of properties only generated at run-time.

For instance, when a product such as a garden tool is treated as an assembly consisting of other assemblies (e.g. motor assembly) and components (e.g. switch), both classes and objects encompass properties that are used to describe their characteristics. The main properties assigned to assemblies and components are shown in Table 4-1. When the aggregate product model is being generated (at runtime), the object properties are assigned values stored in slots that are shown in Figure 4-2 as squares.

In CAPABLE*Assembly* the behaviour of the generation of the objects (components) and their corresponding properties are governed by rules. These rules determine which type of objects need to be created and when they are created, where the received data will be stored, and if the data received are reasonable. They are also used to regulate the interaction with databases. The relationship between the object plane and the rules plane is shown in Figure 4-3.

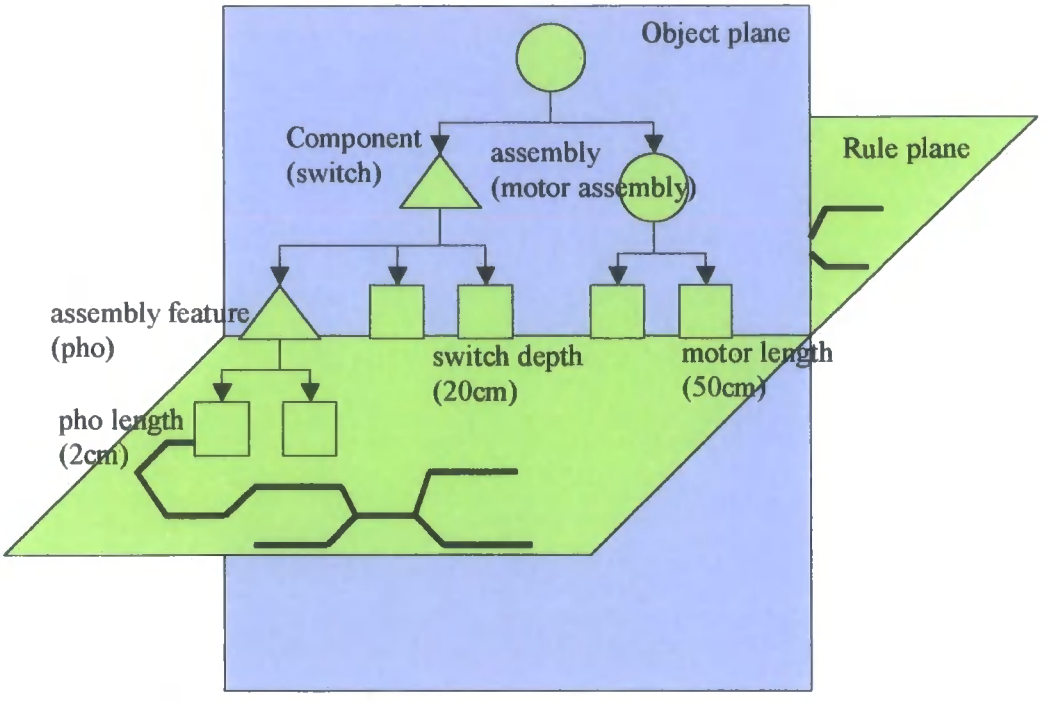


Figure 4-3: Relationship between object plane and rule plane

Classes	Main properties
Product	amount, base_part, breadth, hand_diff, length, no_comps, numafcs, parent, selfname, typeclass, value_alpha, value_beta, volume, weight, width
Assemblies	amount, base_part, breadth, hand_diff, length, no_comps, numafcs, parent, selfname, typeclass, value_alpha, value_beta, volume, weight, width
Components	amount, handling_difficulties, material, nest/tangle, no_comps, numafcs, parent, selfname, typeclass, value_alpha, value_beta, volume, weight
Assembly feature connection	afctype, assembly, assembly_children, base_part, diam_max, diam_min, floating, fixed, handling time, holding, insertion time, moving part, not_align, obstr_access, parallelism, parent, reorientation, restr_vision, selfname, stability, typeclass

Table 4-1: Classes and their properties with CAPABLE*Assembly*

4.2.2 Assembly feature connections

Assembly feature connections (AFCs) refer to the type of mating relationship between two or more assembly features. They allow for the specification of features that connect components together as shown in Figure 4-4. For each AFC considered (see Table 4-2), there exist a series of appropriate assembly methods including initial checks (dimensional analysis) that facilitate the generation of the AFC. The type of AFC generated is dependent on the assembly features used to create it. Figure 4-4 depicts a threaded AFC; a threaded AFC is automatically generated when two threaded features are linked.

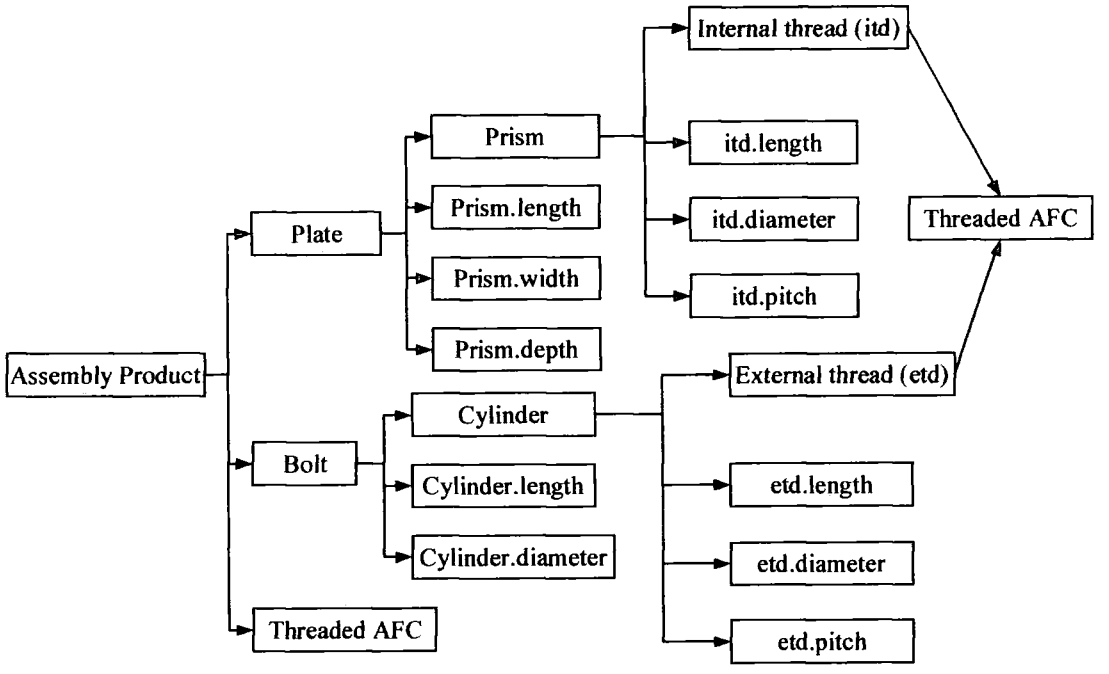


Figure 4-4: Assembly feature connections AFC

AFC objects also store information with regards to the mating parts and their corresponding features, moving and base parts within the AFC, relevant dimensions and handling and insertion data. Handling data provides information with regards to environmental and topological restrictions likely to be experienced whilst performing a specific AFC. Such data include

- Holding down requirements. Holding down is required if parts are unstable after insertion or during subsequent operations.
- Visual restrictions. This applies to parts that are small and parts that are 'blind' to the assembler. For example if one is to inset a peg into a hole that is not within ones visual range.
- Sharp edges. Parts with sharp edges require careful handling thus increasing the time taken to grasp and locate the part within the assembly.
- Nest and tangle. Nesting and tangling in parts occurs when parts are stored in bulk, this increases the time taken to grasp and control the part.
- Weight, size and symmetry. Part weight and size affects the time taken to grasp, control and locate a given part. The symmetry of a part affects the time taken to reorient the part prior to locating the part within the assembly.
- Realignment and re-orientation of moving and base parts. Base parts usually require some means of maintaining the position and orientation of parts already

in place, or during subsequent operations. Moving parts might require reorientation before the can be assembled.

Insertion data solely provides information with regards to the AFCs operation time, including tooling requirements where needed. The classification of assembly feature connections considered during the scope of this research is shown in Table 4-2, but these are constantly being updated.

AFCs Classifications	Assembly Feature Connections	Assembly Feature Connections Sub-Type
Standard Insertion Assembly Operations	Placement	
	Plug 'n' Target	Cylindrical
		Non-Cylindrical
Reversible Insertion Assembly operations	Packaging	
	Pressure Fits	
	Threaded	Screwing
		Bolting
	Wiring	Tag connectors
		Screw connectors
		Pressure-fit connectors
Permanent Insertion Assembly Operations	Adhesives	
	Riveted	
	Thermoplastic Welding	Ultrasonic welding
		Spin welding
		Hot plate welding
		Vibration welding

Table 4-2: Classification of AFCs

The execution of each AFC requires specific tools and assembly workstation equipment. Assembly workstations are described by the processes (AFCs) that they can perform and by other process parameters such as RPM of tools or size of press machines. Hence, a dynamic link is established between the AFCs stored in the aggregate product model and the assembly resources of a factory. The factory model used is described in detail in Chapter 6.

4.3 Connectivity Model

For a given product model, once all AFCs have been created an assembly sequence can easily be generated from the aggregate product model and the BOM of the product using a simple bottom-up assembly approach. That is, within each assembly level the heaviest and/or largest part is chosen to be the base part. The remaining parts are assembled in the order in which they are related to the preceding part within the aggregate product model. At this stage, the assembly sequence generated using such a crude method is at best a feasible assembly sequence.

A sequence planning system requires a model of the logical relationships (surface contacts and attachments), as well as non-geometric information (such as attachment

forces), that are not related to part geometry, but never the less, affect assembly methods (a relational product model). The connectivity model is used to represent relations among the components of assembly, using the aggregate product model as the starting-point of this relational model. The connectivity model adopts the hierarchical representation of the product model as illustrated in Figure 4-5.

When an AFC is created in the connectivity model, two subscript indices are attached to the assembly operation created; $AFC_{i,j}$. The following notation is used in the formation:

i subscript for level at which the AFC is created.

j subscript for the ranking of AFC within the level created.

Each AFC generated within the connectivity model has the following the basic data attached to it:

1. AFC classification. This includes information as to the AFC type, if the AFC is to be regarded as a fixed or a floating AFC (see Section 4.3.2), and if the AFC is to be regarded as a permanent or a reversible AFC (see Table 4-2).
2. Mating components. This refers to the individual components of each AFC as shown in Figure 4-5.
3. Mating features. This refers to the positive and/or negative features attributed to each mating components that are used in the AFC. For example, in Figure 4-4, this would be the internal thread of the plate and the external thread of the bolt.
4. Dimensions of mating components and features. This refers to the length, dept, and width of the mating components. This information is required to calculate the part weight for handling purposes. Dimensions of mating features are needed to calculate assembly variables such as number of threads on a bolt (the equation for a threaded operation is based on the number of threads which is equivalent to the number of turns required to tighten a bolt, see Appendix E).
5. Maximum reorientation angles (Boothroyd and Dewhurst, 1996). This refers to the symmetry of the mating component. The handling time for each mating component is dependent on its maximum reorientation angle, see Section 2.3.1.

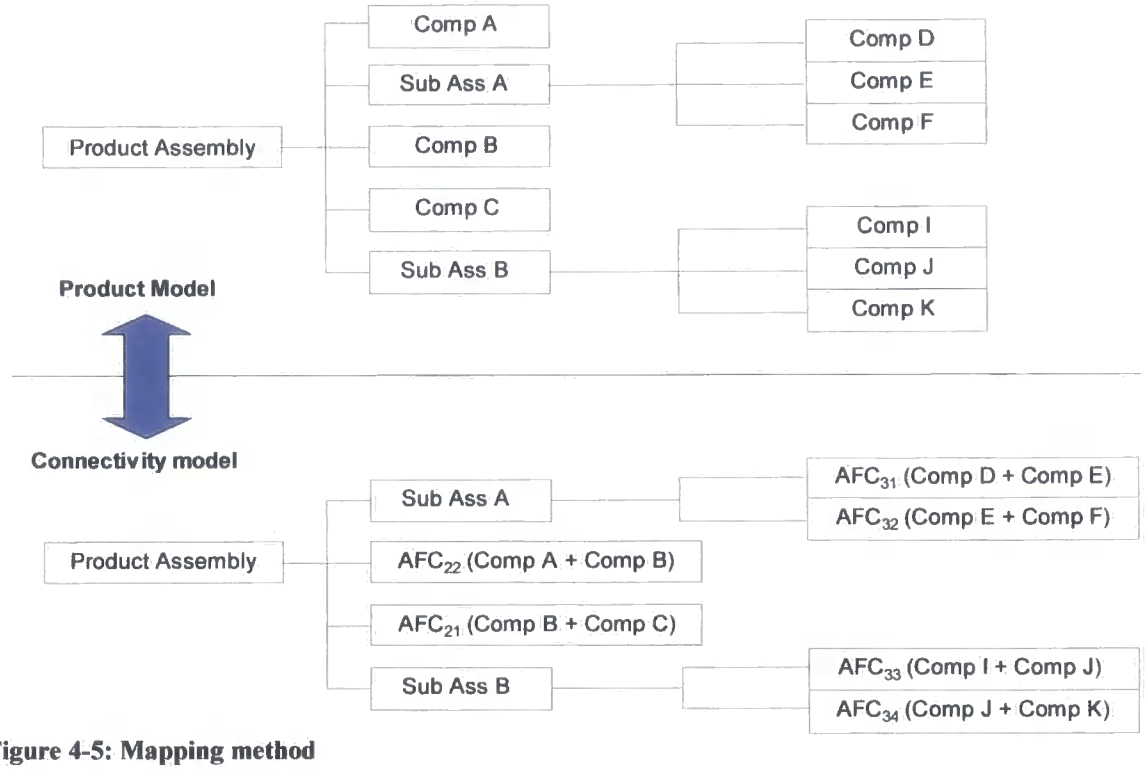


Figure 4-5: Mapping method

The connectivity model like the aggregate product model, are objects created at runtime. Three consecutive algorithms are used to generate the connectivity model namely; contact, precedence and technological constraint algorithms. The aggregate product model is modified after each algorithm; the resulting product model is the connectivity model.

4.3.1 Contact Constraint Algorithm

Contact constraints specify which parts are connected to other parts in terms of an assembly operation, denoted as the AFC's parent. For example, if the external thread (etd, see Appendix D) of a screw is to mate with the internal thread (itd, see Appendix D) of a nut to form a threaded AFC, the parents of the AFC are the screw and nut, denoted as screw_nut. Hence, it is easy to generate a list of AFCs, associated with a given component. Such a list is generated for each component constituting the product. A contact constraint algorithm is used to generate this list; the list is subsequently used as one of the inputs to the precedence constraint algorithm.

4.3.2 Precedence Constraint Algorithm

Precedence constraints represent the fact that some components have to be assembled before others. They are imposed within the hierarchical levels of the product model as well as within subassemblies. A logical set of rules can be derived to establish an adequate ranking system for all AFCs considered. AFCs that are restricted to subassemblies within a given hierarchical level are called "fixed AFCs". An AFC

allowed to move freely between hierarchical levels or subassemblies is deemed to be "floating". This algorithm establishes the relative priorities of the AFCs included in the connectivity model. The algorithm uses the contact list generated in the aggregate product model.

4.3.3 Technological Constraint Algorithm

Technological constraints apply when multiple mating conditions occur in addition to the regular sequential assembly processes. For example, a component may be assembled with more than one component (multiple mating) or there may be a variety of AFCs involved including permanent or reversible AFCs. In such cases the technological constraints are applied to prioritise such operations. For example, permanent AFCs (such as welding and joining using adhesives) are performed after executing all reversible AFCs (such as placement and threaded) of the same level, for a given component assembly. It should be noted that thus applies only to floating AFCs as fixed AFCs maintain their relative positions, and any AFCs that do not follow this precedence law should be made fixed.

4.4 Assembly time Generation Algorithm (AGA)

The assembly time for each AFC object generated (stored as a property of the object subject to handling and insertion data) is calculated for every assembly sequence generated; the summation of the operation times of all AFCs equates to the total assembly time for a given product. The assembly time algorithm is based on the works of Boothroyd and Dewhurst (1996) and the ideology behind Pre-determined Motion Times Systems (PMTS) as discussed in Chapter 2.

The underlying principle in the infrastructure of PMTS is the decomposition of any manual operation or method into the basic human motions required to perform it, and assigning to each motion a predetermined time standard. These so-called "basic motions" are dependent on the system being employed such as Maynard Operations System Times (MOST) and Measured time Motions (MTM). In the original DFA method as devised by Boothroyd and Dewhurst, estimates of assembly times are based on a group technology approach where those design features of parts and products that affect assembly times are classified into broad categories. For each category an average handling and insertion time estimate is established. Clearly, for any particular assembly operation these average times can be considerably higher or lower than the actual times. However, for assemblies containing a significant number of parts the differences tends to cancel so that the resulting total time is reasonably feasible.

The algorithm for the generation of assembly times used herein is an amalgamation of the two methods, coupled with industrial relaxation factors, resulting in the creation of mathematical equations to estimate times for assembly operations by taking into consideration the sequence of the assembly process and the layout of the assembly workplace. It derives an ideal equation for a particular standard assembly operation. These ideal equations are subsequently modified to allow for time constraints to be taken into account. The time constraints imposed on assembly sequences are as a result of the environmental constraints of the assembly workstation in terms of layout and ergonomics. The basic equation for "time constraints" is compatible with experimental data from Boothroyd and Dewhurst. Experimental validation for the method developed is provided in Chapter 7.

A subset of AGA, Standard part assembly methodology (SPAM) can be described as a body of methods used for calculating times for all feasible sequences for a standard assembly operation. The method is similar to that described above, with an emphasis on standard operations using repetitive motions. Here, standard operations are defined as the assembly operations that involve the use of standard parts. Standard operations considered within SPAM include bolting, screwing and riveting. A series of ideal equations are derived for each standard operation using varying body motions and standard part handling configurations. For a given standard assembly operation, an ideal time expression is chosen from a pool of equations depending on time or other environmental restrictions. Mathematical equation, shown in Appendix E, for *tool use* with respect to the standard operations bolting, screwing and riveting have been derived. These tooling equations have been incorporated in the generation of the basic equations, which corresponds to the ideal situations.

A detailed description of the process of assembly time generation for standard parts and standard assembly operations is presented in the following sections.

4.5 The Concept of Standard Parts for Mechanical Assemblies

The use of standard parts in engineering design has been greatly advocated both in design textbooks (Pahl and Beitz, 1984) and by experienced design engineers within industry. The use of common engineering components can easily be found in mechanical assemblies such as gearboxes and engines. These include bearings, seals and springs.

This section introduces the concept of standard parts for mechanical assemblies. It presents the definition and classification of standard parts currently used within sectors

of industry, such as the automotive, aerospace, and electro-mechanical sectors. Also presented is a detailed analysis of the standard part libraries currently available. The importance and essential features required, along with the general structure for a standard parts assembly database are subsequently discussed. Finally, an extract of the current system is presented.

4.5.1 Definition and classification of Standard Parts

In general, standard parts can be defined by part functionality. Most common part functions include bearings, fasteners, gears, seals, shafts, springs and wires. Within each function there exist a range of core components that require a particular assembly sequence or deformation during the assembly process. Such components or parts are deemed to be standard parts.

The use of standard parts has been classified (Betteridge, 2000) within three broad types of assembly, namely mechanical, electro-mechanical and electronics, shown in Figure 4-6. Table 4-3 shows examples of standard parts within the above-named assembly types.

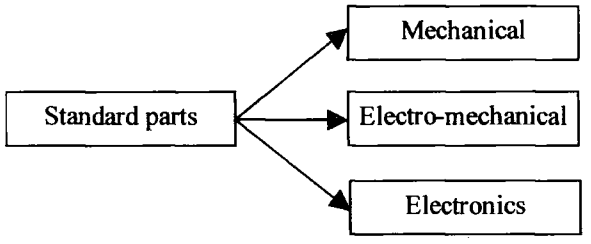


Figure 4-6: Classifications of Standard parts

Classifications	Examples of Standard Parts
Electro-mechanical	Motor, batteries, and generators.
Electronics	Capacitor, cables wires, fuses resistive products, and diodes.
Mechanical	Bolts, screws, nuts, washers, rivets, pins, springs, and bearings.

Table 4-3: Examples of Standard parts

4.5.2 Why Create a Standard Parts Database?

The advantages of using standard components are numerous. Pre-defined reliable physical and performance characteristics, favourable supply conditions (which can be set-up to aid production) and cost effectiveness are just a few of the benefits reaped from the use of standard parts.

A standard parts database is included in CAPABLE*Assembly* to aid the process of product specification and to derive accurate assembly times using as little input from the

user as possible. The standard parts database contains information with regards to geometric and parametric descriptions of parts, as well as other required assembly data (e.g., manipulation data required for the calculation of assembly process times). The objectives of creating such a database are:

1. Reduce complexity by limiting user input: an increase in the amount of information required from the user will only serve to increase the complexity of a DFA system. Using standard parts will decrease the data required from the user, as a selection system will be established. Reducing the complexity of the system will increase the overall efficiency of the system. The use of stored data lends itself to the reduction of computing time and available resources.

2. Provide a comprehensive parts list in standard format, that is, ISO, DIN and BS. Not only do sub-assemblies have to adhere to a particular country's standard, it is also imperative that the product specifications provided falls within the guidelines as set by the respective standards. All geometric and parametric descriptions of standard parts covered in the database are according to the well established governing bodies namely, International standards organisation (ISO), British standards (BS) and German standards (DIN).

3. Minimise computational time and resource requirements: The computational time of CAPABLE*Assembly* will be greatly reduced by the adoption of a standard part database. The database is capable of performing queries and producing summaries on particular product groups. Such a facility can only serve to minimise computational time as well as resources requirements.

4. Aid the calculation of realistic assembly times: The use of standard parts will ultimately result in the provision of more realistic assembly times.

When considering mechanical assemblies, the main joining operations performed include; fastening by screw or bolt, riveting, pressing, and welding. Indeed, fastening by screws or bolts together with riveting constitute over 60 percent of all mechanical assembly operations. With these facts in mind, the database has been designed specifically for the component category of FASTENERS. In the following section, the process of deciding which parts would be included within the database, the type of product information required and the database functionality's will be presented.

4.5.3 Data Acquisition

Prior to designing and setting up a database, a process of information gathering had to be undertaken. For the product group of fasteners, a list of fasteners was initially generated using the following paper-based components catalogues:

1. Hpc Gear, Transmission Catalogue
2. Hpc, Mechanical driver components Catalogue
3. RS Components Catalogue - Mechanical
4. SPEC, Stock Precision Engineering Components products catalogue

The referenced catalogues are non-manufacturer specific. This provided a good all-round view of the parts and tools currently available in the targeted market.

Further investigations were made into obtainable tools and parts using manufacturer catalogues. This was mainly done in the form of on-line company catalogues namely, Black and Decker, Bosch, GESPIA, International Fasteners, K&L Fasteners, Lobster Tools and SKIL Power tools. Due to the wealth of information received in terms of tool availability, a sorting process was applied to eliminate parts and tools that were not to be considered under the given definition and classification of standard parts within mechanical assemblies. Furthermore, parts that were initially deemed to be standard parts but did not met the BS, ISO, or DIN standards were identified.

With the information left, a process of compiling the data was executed. Products were grouped according to their product type and product name, part size, and part thickness. Possible handling times for each were also obtained for each standard part using the Boothroyd and Dewhurst time standards, as shown in Figure 4-7. The parts are expressed in terms of assembly modelling method described in section 4.2.1, as shown in Figure 4-7.

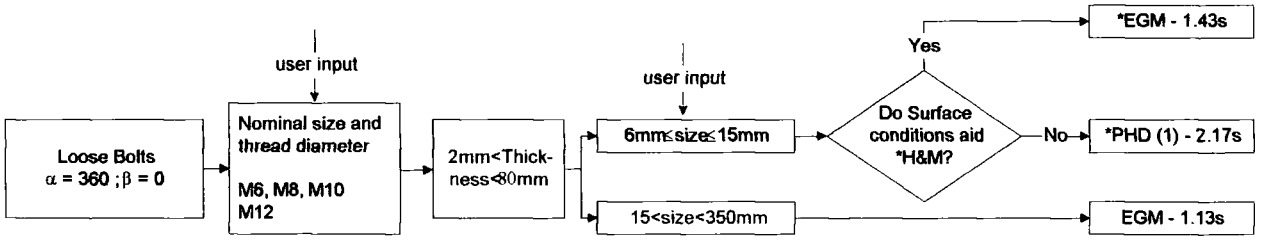


Figure 4-7: Data acquisition

4.5.4 Standard Parts Databases for Mechanical Assemblies (SPAD)

A standard part database for mechanical assemblies, "SPAD" has been created for CAPABLE*Assembly*. Due to time limitations the part forms shown here have not been

integrated in CAPABLE*Assembly*. However, the corresponding flat file database currently used to store all the standard parts considered is accessed directly by CAPABLE*Assembly*. The data currently available within SPAD has been limited to mechanical fasteners such as bolts, nuts, washers and rivets.

The database has been specifically designed to aid a design engineer when using DFA methods. The database has been designed for use at the aggregate level of product design. All products listed within the database are stored in accordance to the aggregate product model.

A number of computer systems for specific components have already been developed. Most of the software packages widely available are for mechanical engineering components. However, there are a limited number of packages becoming increasingly available for electrical and electronic components as well as materials and adhesives. Of the mechanical engineering components, bearings have received considerable attention.

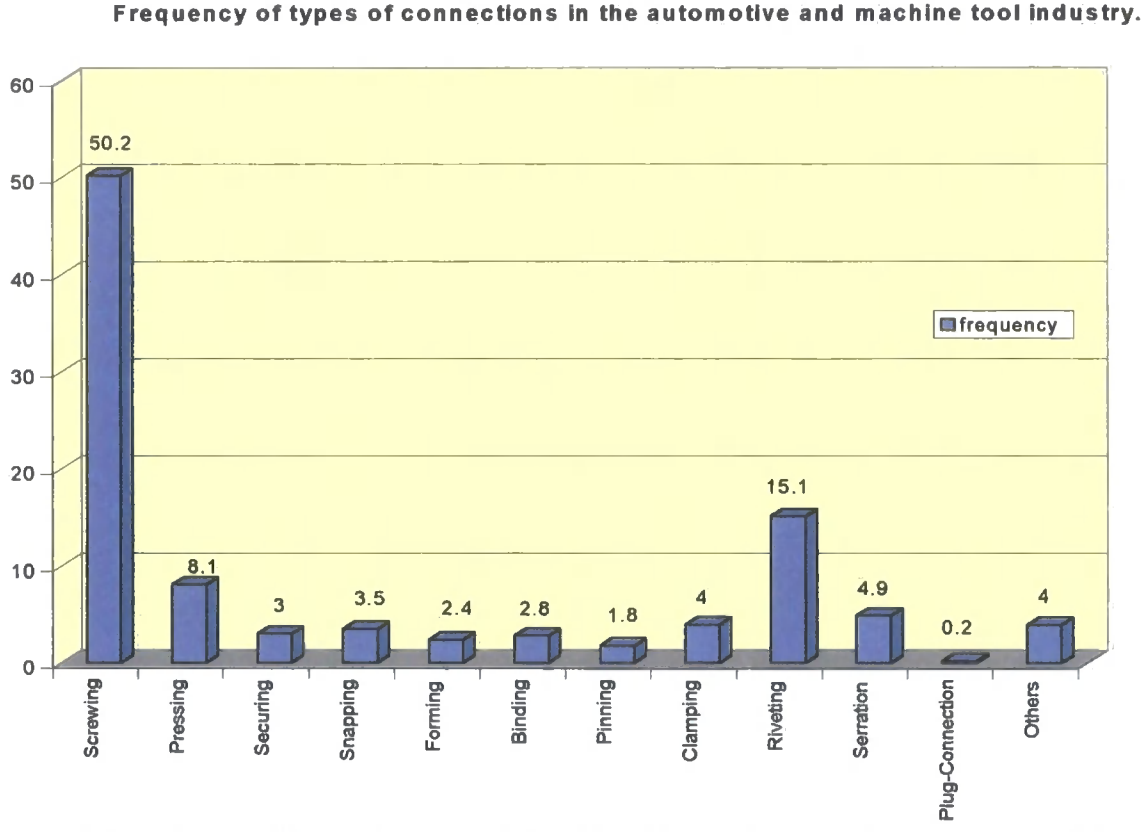


Figure 4-8: Frequency of types of connections in the automobile and machine tool industry (Abele, 1984; Gießner, 1975; Schweizer, 1978; Warnecke and Walther, 1984)

The basic work with regards to the development of a standard part assembly database for mechanical assemblies is the customisation of such packages described above to suit the needs of a design engineer using DFA tools at the conceptual stage of design. Hence, it was decided that for the purpose of generating standard parts databases for mechanical assemblies, fastening would undoubtedly be the most logical function to be

implemented. The frequency of such types of connections in the automobile and machine tool industry is shown on Figure 4-8. For the purpose of this research, the assembly connections considered are bolting, screwing and riveting. As can be seen from Figure 4-8, this constitutes over 60% of assembly operations within the mechanical engineering industry.

There are various methods of joining components. Assembly connections of product components can broadly be divided into two sections, permanent joining and temporary joining methods. Examples of permanent joining processes include welding, soldering and use of adhesives. Temporary joining techniques comprise processes such as screwing, bolting and riveting.

As it is now commonplace for design engineers to use advanced CAD systems, the development of computer aided component selection system will prove to be a very useful tool. The advantages of such software systems has been well documented, they include:

1. A much-improved likelihood of identifying the best available component.
2. Speeding up of the selection procedure.
3. All component types and variants can be considered without prejudice.
4. A common selection procedure can be used for all available component variants (especially true for when a windows operating environment is used).
5. The use of 'standard' catalogue components is encouraged as opposed to 'specials'.
6. Accuracy and reliability of results are assured.
7. Presentation quality of calculations is improved.

4.5.5 Fields within SPAD

The standard parts gathered (see Section 4.5.3) are stored in Microsoft Access. Fields are used to store vital information for each part. The naming conventions of Microsoft Access were followed when naming the fields within SPAD. A field is defined as an element of a table that contains a specific item of information, such as a product name, supplier's names and manufacturers specifications. A field is represented as a column or cell in a data sheet and as a control on a form. A field is defined by entering a field name, a data type, and a description (optional). The set of data types available to choose from for a field in Microsoft Access includes counter, currency, date/time, memo, number, OLE Object, text, and Yes/No properties.

Fields in SPAD include; product ID, product type, product name, product features, alpha and beta angles, and parametric definitions of parts. Other fields that will be included in due course include values of pitch for threaded standard parts, weight of standard part (this will ultimately lead to the development of a linked materials database) and the tooling resources currently available for operations concerning the use of a particular standard part.

1. Product ID: The Identification of a product within the database is dependent on the part function. Within mechanical assemblies, part functions include bearings, seals, pneumatics, plumbing and pipe work. It has already been stated that SPAD will deal with all standard parts under the part function of fasteners.

   The product ID is used to define which functionality the part performs. At the moment, all parts within SPAD fall under the part function of fasteners. In due course the product IDs within SPAD could be extended to cover other part functions within the mechanical assemblies. The data type input for this field falls under the category of "Text". The product ID can be utilised as a base for running queries and producing reports for a particular database. For example, the database can be used to generate a summary of all parts under the product ID of fasteners. Such a task can easily be performed using Microsoft Access. The output of such an assignment is usually in the form of a report, which can be exported as a text file.

2. Product Features: The product features defined utilise aspects of form features, design features and manufacturing features. Product features are categorised under two broad headings namely, positive and negative features. Positive features describe a geometric shape, which encloses material volume. Examples of positive features are cylinders and prisms. A negative feature describes a geometric shape from where material has been extracted. An example of a negative feature is a hole drilled through a prism.

   It has already been discussed that at the aggregate level, the overall description of parts or features should be a sufficient geometric representation of the part. The data type allocated to this field is as above, text. Feature definitions along with positive, negative, axisymmetrical and prism features have already been discussed. A list of all features currently in use can be found in Appendix D.

   As an example, the feature definition of a plain nut is show in Figure 4-9.

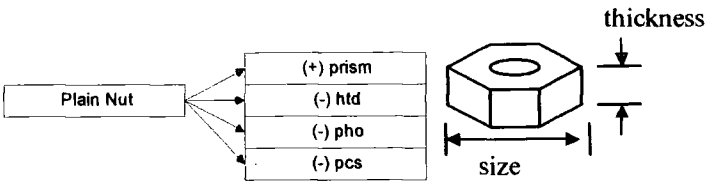


Figure 4-9: Product features of a plain nut

The positive feature of a plain nut is defined as a prism. Negative features are a through hole and a tapered entrance. It is important to note that feature definitions do not aim at fully defining a product, as this information is not required for manufacturing purposes.

3. Product Type Classification: This refers to the general type of a standard part found within a functionality class. For example, for the part function fasteners, there exists a range of parts such as bolts and nuts, which can be considered as a product type. The different product types under fasteners are shown on Figure 4-10.

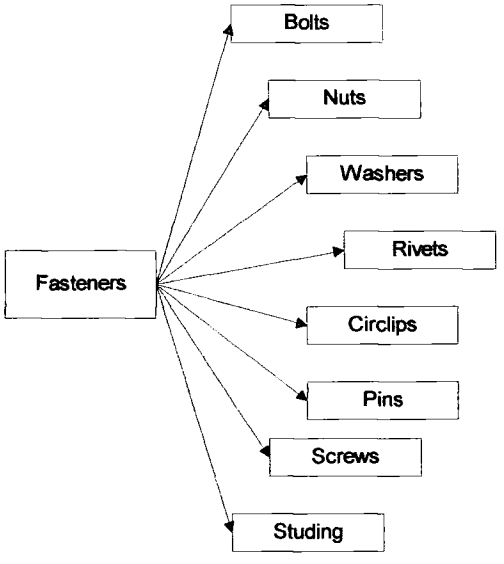


Figure 4-10: Fasteners; Product type Classification

It has already been stated that standard parts can be defined by their part function. Each part consists of a range of products, which are categorised in terms of their type. For example, the decision as to which class of parts falls under the umbrella of fasteners was made by investigating the range of components supplied by companies that specialised in the design, production and retail of fasteners. Design catalogues were also utilised. These including RS Catalogue, K & L Fasteners, Industrial Fasteners, Lobster Tools and PM Speciality Fasteners. For a complete selection of products and product types for

fasteners, see Appendix F. Products found in Appendix F include Bolts, Screws, Washers, Pins, Nuts, and Rivets.

4. Product Name: Each product type may comprise several different sub-types of standard parts associated with the particular part classification. The data type for this field is text. The sorts of data input in this field include terms typically used within industry when referring to different types of standard parts within a certain class. Figure 4-11 shows the sub-class of the "bolt" class of fasteners. A complete description for all classes of parts considered can also be found in Appendix F.

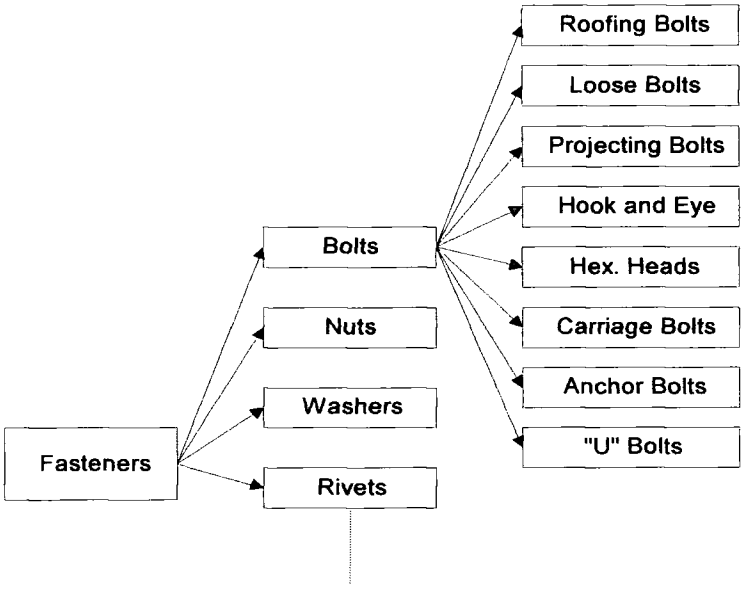


Figure 4-11: Fasteners; Product Name

5. Alpha & Beta Angles: This is a direct application of Boothroyd and Dewhurst manipulation angles. Alpha and Beta angles have already been discussed in Section 2.3. The data type is numerical. Alpha and Beta values are typically the same within a given product class. Maximum orientation angles refer to the alpha and beta angles of symmetry as defined by Boothroyd and Dewhurst, for product handling purposes. Hence, each product type has been assigned an appropriate alpha and beta angle of symmetry.

6. Parametric Descriptions: Refer to the critical dimensions that fully describe a standard part. Critical dimensions of interest for handling purposes include the length/size and diameter/thickness of standard parts. The range of standard parts covered is with accordance to suggested dimensions and material properties by British Standards. This is a numerical data field. It will contain parametric

details defining a particular standard part. Parameters typically include thickness and size.

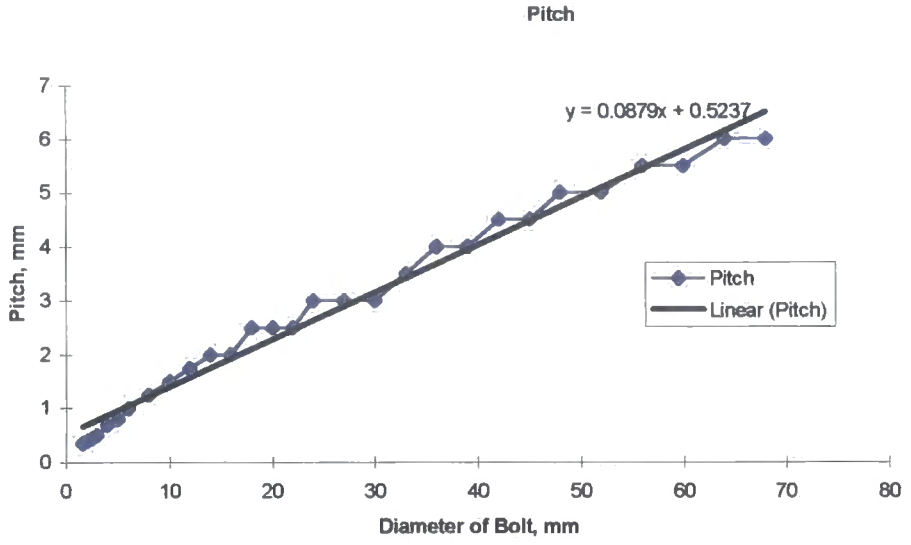


Figure 4-12: Relationship between bolt diameter and pitch

Linear trends were obtained using spreadsheet packages, which resulted in the parametric descriptions for a range of standard parts. This results in a parametric expression as a function of the parts' critical dimensions (for example the diameter of bolts, thread size). Figure 4-12 shows the relationship between bolt diameter and pitch size.

Where deemed necessary, the weight of parts will be represented as a function of the parts' critical dimensions and material properties. However, it is hoped that such data will eventually be obtained from a CAD system. Data can be stored in two ways, in the form of a table or as a chart.

7. Tooling resources: Although not fully functional within the database, tool types were selected to cover a wide variety of tools utilised when an assembly operation is performed on a product type. At present, data purely consists of fastening and loosening times and do not include handling times for the tool depending on the assembly sequence. For example, Table 4-4 shows the fastening times for several riveting tools. For a comprehensive list of tooling resources and fastening data see Appendix E.

Tool Type	Rundown time /Revolution
Lazy Long Type	2.06 sec/stroke
Standard Rivet Gun	1.03 sec/actuation
Hydro-pneumatic Riveter	1.33 sec/actuation
Manual Lever Riveter	1.53 sec/actuation

Table 4-4: Riveting Tools

4.5.6 Extract from SPAD

An extract of product classes "bolts" and "nuts" is shown below. The extracts are presented in the form format. Figure 4-13 shows the sub-class of anchor bolts.

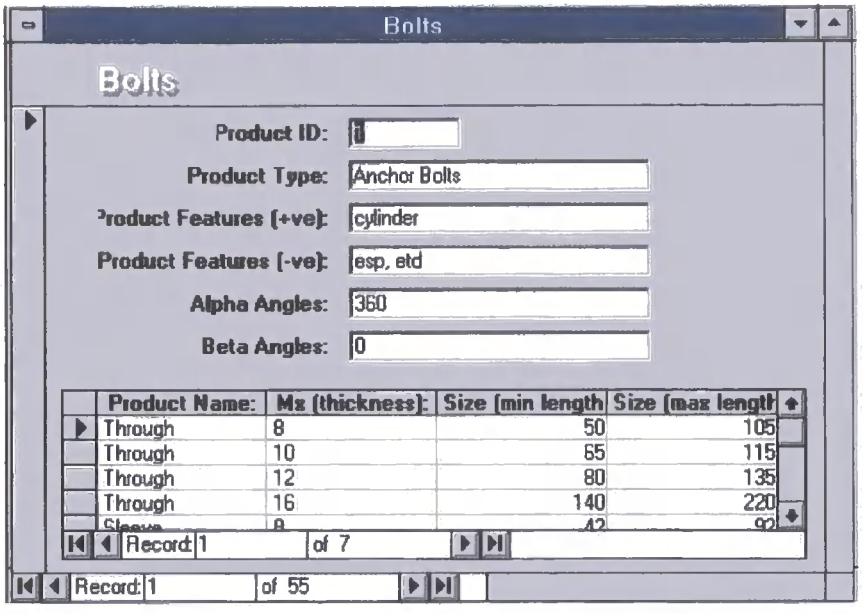


Figure 4-13: Product Class: Bolts, Sub-class: Anchor bolts

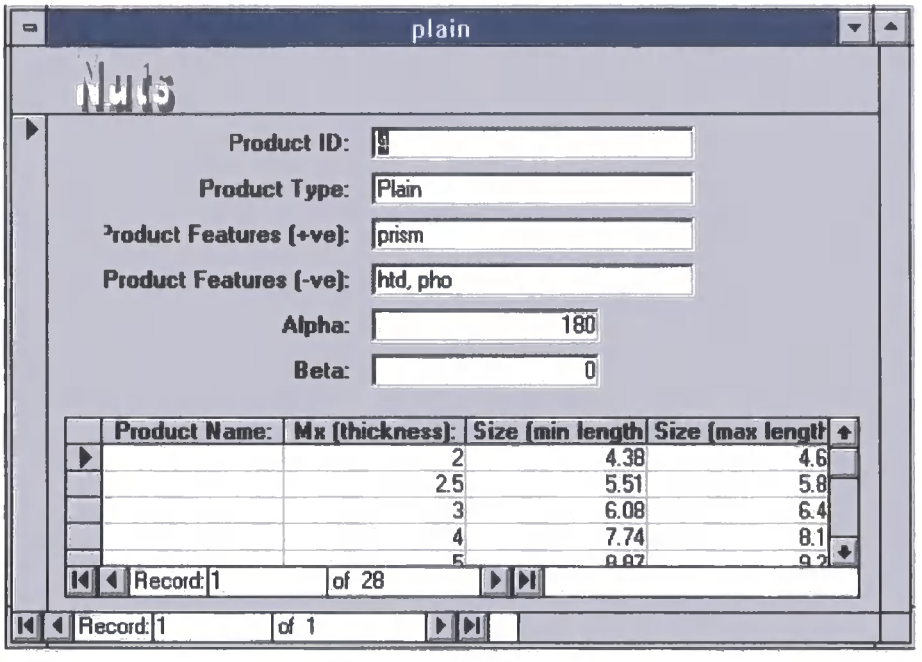


Figure 4-14: Product Class: Nuts, Sub-class: plain nuts

Positive and negative features, alpha and beta angles can clearly be seen. The scroll situated to the left of the sub-class allows the user access to the range of anchor bolts stored within SPAD.

Figure 4-14 shows the sub-class "plain nuts" (belonging to the class "nuts"). Positive and negative features, alpha and beta angles can clearly be seen. The sub-class "plain nut" does not contain further sub-classes.

An extract of the product class "tools" is shown in Figure 4-15. The extract is presented in the form format. Within this class, tooling information with regards to riveting and screwing/bolting is stored. All tools are stored within the same class. The assembly operation along with its respective tool can be found on each form as shown in Figure 4-15. The arrows at the bottom of each form allow the user to browse through the class. Figure 4-15 shows the form for a riveting tool.

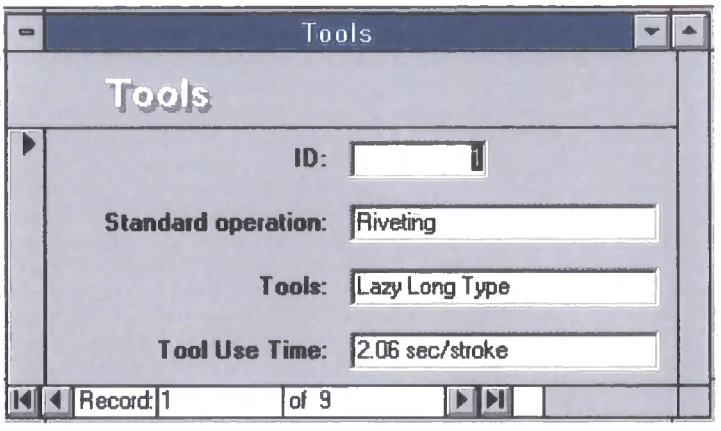


Figure 4-15: Product Class: Tools, Standard operation: Riveting

4.6 Standard Part Assembly Methodologies (SPAM)

4.6.1 Introduction

In assembly work, the efficiency of the process is greatly affected by the sequence of assembly. With regards to assembly sequences, the number of alternatives tends to be large. Consequently, the process of selecting the most cost-effective assembly sequence from a large set of alternatives is a difficult and important task faced by design engineers.

Standard part assembly methodology is used within the CAPABLE*Assembly* to aid the process of assembly modelling and to derive accurate assembly times for standard assembly operations. Mathematical expressions for estimated assembly times from standard assembly operations are also derived.

Standard Part Assembly Methodologies (SPAM) can be described as a body of methods used for calculating assembly times for all feasible sequences for a standard assembly operation. It has already been stated (Figure 4-8) that manual joining tasks such as bolting and/or screwing and riveting account for more than 60% of mechanical assembly connections performed within the automotive, aerospace and machine-tool industries. This indicates the importance of modelling and optimising fastening operations. The SPAM set of techniques can be used to generate expressions to estimate assembly times for mechanical assembly connections and evaluate various feasible

sequences. Both the Standard Parts Assembly Database and the Standard Part Assembly Methodologies are an extension of previous work done on standard parts by Betteridge (2000).

In this Section, the concept of standard assembly operations is presented. SPAM is subsequently applied to chosen standard operations. An explanation of the methodology is presented along with an explanation of the modifications to elements of the two DFA systems utilised namely, Boothroyd and Dewhurst DFA method and Maynard Operational Systems Technique.

Finally, an example of how the method can be used to estimate the assembly time of a product is presented. Possible extensions of the methodology to other assembly operations are also discussed.

4.6.2 Standard Assembly Operations

The establishment of assembly connections is one of the most common assembly tasks in design, and the sequence of such connections a central problem which has to be considered in assembly rationalisation.

The concept of standard assembly operations has been developed at Durham University as a natural progression from the concept of standard part libraries (Betteridge, 2000). Initially, standard assembly operations were defined as operations linking together a number of custom parts using joining methods typical of assembly processes.

Standard operations can be regarded as standard AFCs, where the assembly connection requires the use of standard parts. The concept of standard operations is best described using a bill of materials of an arbitrary product, as shown in Figure 4-16.

The concept of Standard assembly operations developed is now best described as assembly operations linking together a number of custom parts by using an assembly process which involves the use of standard parts and their respective tools. An example of such assembly connections can be seen in Figure 4-16. The generic form that such connections will take place can be described as "screwing". Other such connections include bolting, snapping and riveting.

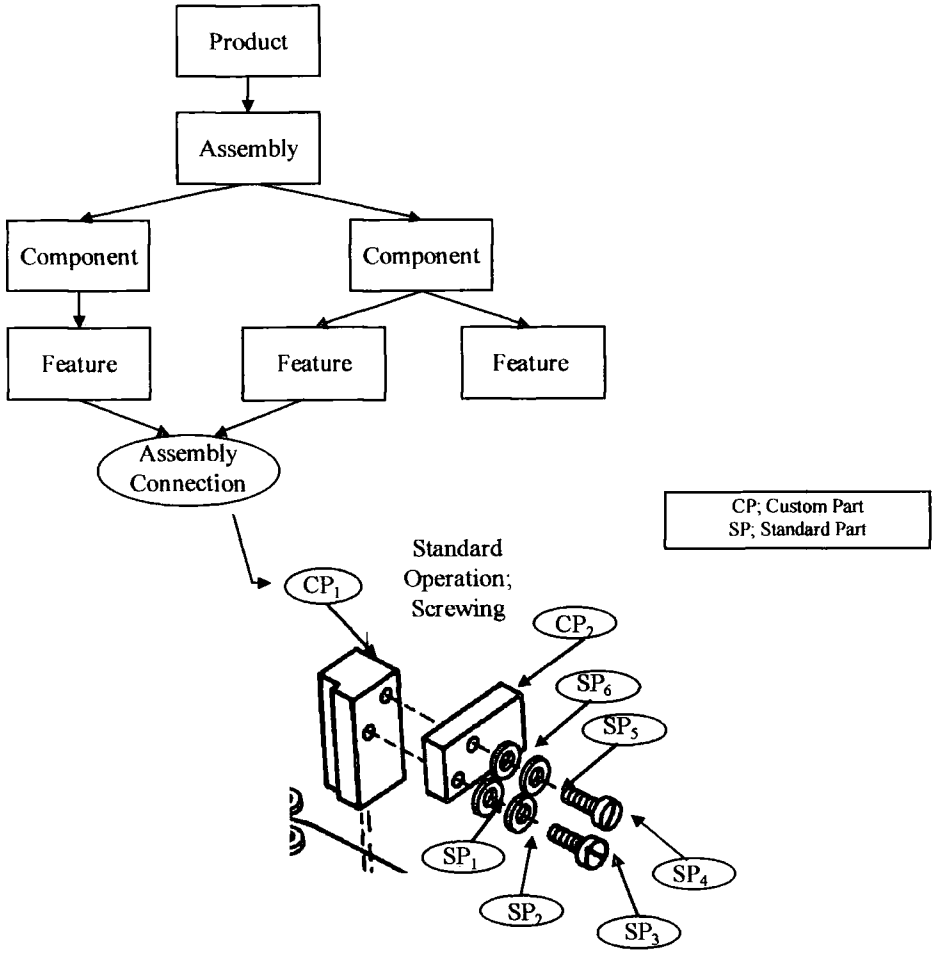


Figure 4-16: A Standard assembly operation

4.6.3 The SPAM Methodology

The next step, the process of assembly modelling and the generation of assembly times within CAPABLE*Assembly*, is to develop a method or a group of methods that can be applied to all standard assembly operations with the ultimate aim of deriving expressions to estimate assembly times for standard assembly operations, taking into consideration the sequencing of the assembly process and the layout of the assembly workplace.

At the conceptual stage of design, it is likely that there will be insufficient information to obtain accurate assembly times. With the realisation that the majority of assembly operations performed fall under the umbrella of standard assembly operations, it is feasible to use SPAD to calculate times for any standard assembly operation considered. The resulting methodology developed is an amalgamation of Boothroyd and Dewhurst assembly times and the basic theory behind Pre-determined motion times.

The method has been applied to a number of standard operations namely bolting, screwing and riveting to obtain varying time expressions for assembly times for the named standard operations. The base of the methodology is an adaptation of MOST

SYSTEMS (MOST SYSTEMS, User Manual, H.B. Maynard and Company, Inc.). It utilises the notion that basic assembly motions can be broken down and codified. The basic difference between SPAM and MOST systems is the incorporation into SPAM of the Boothroyd and Dewhurst time constraints on (as discussed in Section 2.3);

- Weight
- Alignment and realignment
- Holding down
- Obstructed access

SPAM utilises the time values obtained from Boothroyd and Dewhurst experimental work to obtain modifications to the existing parameter index values provided in the current version of MOST.

SPAM derives time expressions by breaking down the assembly actions. The terminology (as described in Section 2.9.3) used to describe assembly task in MOST, is adopted, and extended. SPAM derives an ideal expression for a particular assembly operation, and time constraints are subsequently added to the basic time derived. The time constraints are dependent on the layout of the workplace and the application.

The first step taken when developing assembly time expressions using SPAM is to observe the entire standard operation. Once the entire operation has been observed, the basic assembly motions are expressed using a simple flow chart like format, as shown in Figure 4-17.

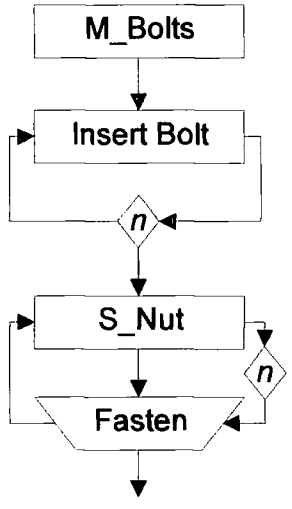


Figure 4-17: Bolting; Sequence BN_1

The entire operation is converted to a representation using an adaptation of the MOST SYSTEM sequence models. The modifications included to the current parameter index values of MOST are explained in the following section. SPAM currently only utilises the General Move Sequence and the Tool Use Sequence. Each assembly time expression for a particular sequence of a standard operation is initially generated for ideal situations, as described in the flowcharts, as shown in Figure 4-17.

Where,

M_Nut is interpreted to mean 'Collect a handful of bolts from a parts bin.

S_Nut is interpreted to mean 'Collect a single of nut from a parts bin.

Repeat process n times.

A complete list of all flow charts and derived equations can be found in Appendix E.

Time penalties are subsequently added on to the basic equation. The expressions for time penalties have been extracted from Boothroyd and Dewhurst experimental data. The time penalties imposed on assembly sequences are as a result of the environmental constraints of the assembly workstation in terms of layout and ergonomics.

4.6.3.1 Preliminary Activities

Three types of sequence models are needed in MOST to measure manual work, and three additional for heavy work, using material handling equipment. Of particular interest are the "General Move" sequence and the "Tool Use" sequence. These sequences have already been detailed in Section 2.9.3. The general move sequence has four sub-activities to account for distinct situations, A, B, G and P. The activity groups along with the respective sequence models are shown in Table 4-5. Time related index values based on the motion content of the sub-activities are placed on each sequence model parameter. The sequence model for both the General Move Sequence and the Tool Use Sequence is shown in Table 4-5.

Manual Handling		
Activity	Sequence Model	Sub-activities
General Move	ABGABPA	A; action distance, B; body motion, G; gain control, P; Place
Tool Use	ABGABP (F/L/M/R/S/T) ABPA	F, fasten; L, loosen; M, measure; R, record; S, surface-treat; T, think

Table 4-5: MOST activity sequences shown with the sub-activities (MOST Systems, H.B. Maynard and Company, Inc., User Manual).

Prior to any modifications to the MOST system, it was deemed necessary to investigate the correlation between MOST and Boothroyd and Dewhurst DFA method. A simple test would be to estimate the assembly time for a given standard operation using the two methods. A positive result is defined as assembly times within an error margin of ±5% for the given assembly operation.

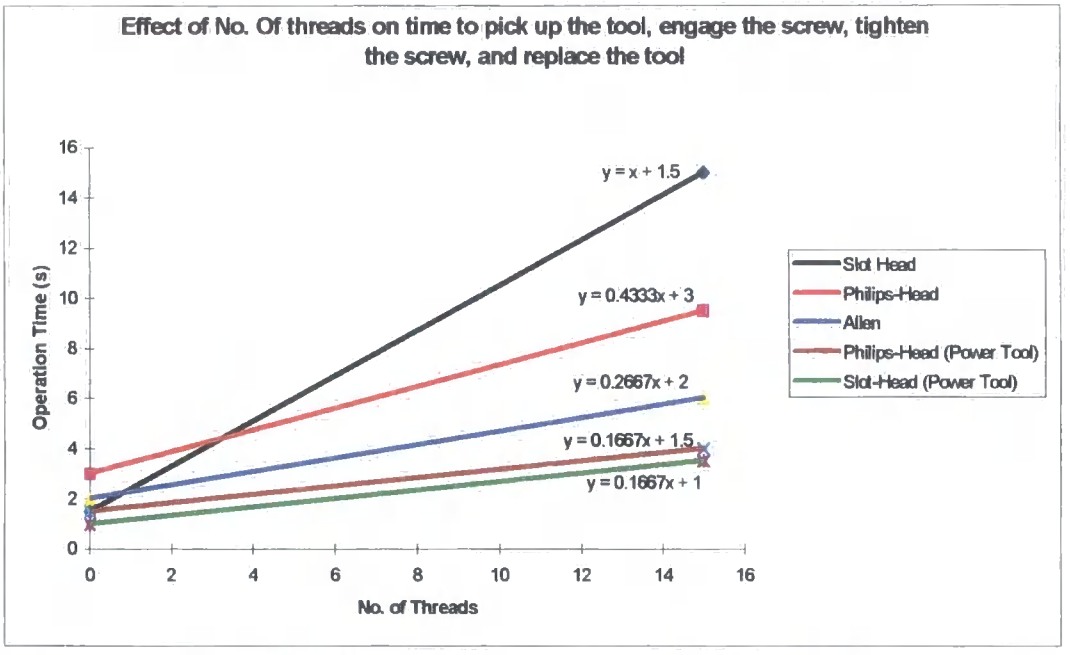


Figure 4-18: Operation times for screwing (Boothroyd and Dewhurst, 1996)

From Boothroyd and Dewhurst experimental work it is possible to obtain actual times for a number of assembly operations, as shown in Figure 4-18.

For the purpose of this analysis, it is imperative that the sequence of the standard operation chosen can accurately be described using both MOST and Boothroyd and Dewhurst DFA method. Consider the following cases using both MOST and Boothroyd Dewhurst DFA method.

4.6.3.2 Case Study 1

Estimate the time taken to fasten a slot-head screw when a screwdriver requires six turns.

1. Using MOST System

   Tool Use Sequence with associated parameter index values:

   $A_1\ B_0\ G_1\ A_1\ B_0\ P_1\ F_{161}\ B_0\ P_1\ A_0$

   $(1+0+1+1+0+1+16+0+1+0)\ 10 = 220$ TMU

   Which is equivalent to 7.9 seconds.

2. Using Boothroyd and Dewhurst DFA method

   From Figure 18 the time taken to pick up tool, engage screw, tighten the screw and replace the tool is:

   $y = x + 1.5$

   Where y is the operation time and x is the number of threads. The number of threads is equivalent to the number of revolutions or turns needed to tighten the screw.

   6+1.5 =7.5 seconds

4.6.3.3 Case Study 2

Another check performed involves the use of Boothroyd data to generate parameter index values. The time values used in the Boothroyd and Dewhurst DFA method were the result of extensive experimental work and work-study cases. MOST assign index values by initially observing the procedure and by subsequently attaching a corresponding index value within approved TMU intervals. Hence, if the time or an expression for a specific operation can be deduce using validated experimental data it should be feasible to calculate the index value for a particular sub-activity.

Using the data presented in the case study above (case study 1), the total operation time using a slot head is 7.5 seconds. According to Boothroyd data, 1.5 seconds of this time is spent picking up and replacing the tool (in Figure 4-18, the tool operation time corresponds to the slope of the graph).

Using MOST systems,

Obtain tool: $A_1 B_0 G_1$

Place tool on screw: $A_1 B_0 G_0$

Relinquish tool: $A_1 B_0 G_1$

Total parameter index = 6, convert to TMU (x10) = 50 TMU.

Convert 50 TMU to seconds (÷ 27.3) = 1.8 seconds

From the results of the case study, it can be seen that the Boothroyd DFA methods gives lower assembly time estimations when compared to the MOST system. The difference in estimated assembly time for case study 1 is 0.4 seconds, and that of case study 2 is 0.3 seconds. As a result, for the purpose of this research, when mapping from the

Boothroyd DFA methods to MOST, a maximum percentage difference of 5% has been added, and vice versa.

4.6.4 Modifications to MOST & Boothroyd Dewhurst Design for Assembly Method

The following modifications were made to the MOST system's basic sequence models:

1. The MOST SYSTEM regards part orientation as "adjustment or fumbles", and can be found under the placement sub-activity. These adjusting motions fall within four basic types, which are determined by the part's relationship to the axis of insertion/placement angular, lateral, and rotational and insertion as shown in Figure 4-19.

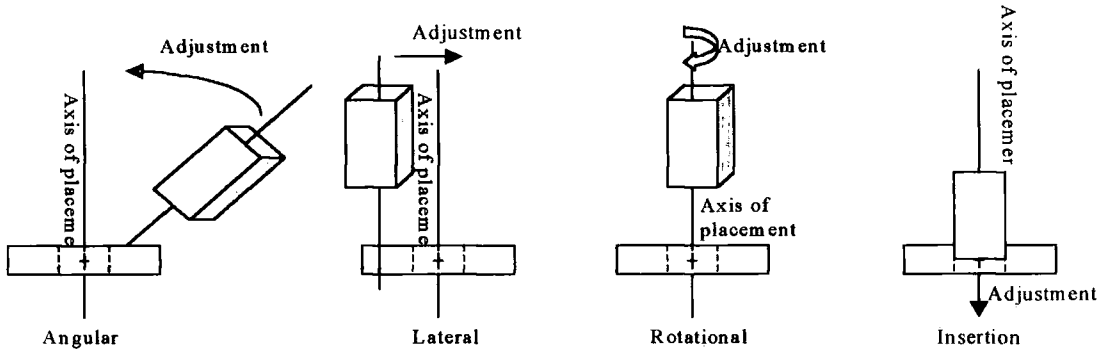


Figure 4-19: The four basic types of adjustments defined by object relationship with axis of placement

In the case of Boothroyd and Dewhurst DFA method, the effects of part orientation have been investigated. The AAMP system currently utilises Boothroyd and Dewhurst DFA type data. In comparison, the codification for part orientation within MOST is insufficient. For this reasons it was deemed necessary to adopt an index value to MOST to account for time differences in handling parts. Body motions prior to the placement of the part have been included within SPAM. Initially, the codification of handling of parts given the orientations listed in Table 4-6 was $A_0\ P_3\ A_0$.

Orientation Angles as defined by Boothroyd Dewhurst	Estimated part handling time obtained from Boothroyd Dewhurst Time standards (Sec)	Resulting Equivalent Index Values Utilised within SPAM
$(\alpha+\beta) < 360$	1.13	$A_0\ P_3\ A_0$
$360 \leq (\alpha+\beta) < 540$	1.5	$A_1\ P_3\ A_0$
$540 \leq (\alpha+\beta) < 720$	1.8	$A_3\ P_3\ A_0$
$(\alpha+\beta) = 720$	1.95	$A_3\ P_3\ A_0$

Table 4-6: Modifications to part handling

2. The MOST system currently accounts for the difference in obtaining one object from a part bin, and obtaining multiple parts from a parts bin. MOST, however

does not account for the effect of part thickness and size on handling time. From the Boothroyd and Dewhurst work, expressions for the effect of part thickness and size on handling times can be extracted. Also, corresponding MOST index boundaries can be obtained from the user manual together with the equivalent time boundaries in seconds. To obtain the different index values that were dependent on part size and thickness the Boothroyd and Dewhurst data were utilised. Prior to this it was checked that MOST time values were comparable to Boothroyd and Dewhurst. Using the Boothroyd and Dewhurst handling data, the handling time for a part that:

- presents no handling difficulties
- is easy to grasp and manipulate
- has a thickness greater than 2mm
- has a size greater than 15mm
- has an alpha[1] and beta values within the range of 360° and 540°

is 1.43 seconds. The equivalent time using MOST systems is 1.43 seconds. Again, this shows the results are within a five- percent error band. The derived mappings is shown in Table 4-7.

Part Size as defined by Boothroyd and Dewhurst	**Estimated part handling time obtained from Boothroyd Dewhurst Time standards (Sec)**	**Resulting Equivalent Index Values Utilised within SPAM**
size < 15mm	1.5	$A_0 P_3 A_0$
6mm ≤ size < 15mm	1.8	$A_1 P_3 A_0$
15mm ≤ (α+β)< 80mm	2.25	$A_3 P_3 A_0$

Table 4-7: Modifications to MOST

3. As a result of validations performed on the derived expression using the SPAM methodology it was found that it is necessary to include a palming action parameter index if more than two parts were obtained from a part bin in a single motion. This created an additional parameter index value P_3 (1.1 seconds). This value is required when grasping and controlling multiple parts. For example, if an operator picks 6 bolts from a parts bin, he/she will require a palming action to reorient the part before insertion.

4. SPAM includes actions that do not belong to any of the sub-activities presented in the basic MOST, in particular run-down time for nuts. A parameter and

[1] MOST as does MTM uses a maximum value for beta angles, but does not take into consideration alpha values, hence the total angle of symmetry was chose to be between 360° and 540°.

suitable codification has not been allocated to these actions within MOST. Times for such actions have been extracted from the Boothroyd and Dewhurst experimental data. These times are simply added on once the entire sequence has been converted from TMU to seconds.

5. *Tooling Sequences:* In the process of deriving time expressions using the Tool Use sequence as presented in the MOST (basic) user manual, it was found that there were insufficient listings of tools. Also for the tools listed, the information presented did not fully define the basic assembly motions observed during the validation process. For the tools listed, the development of the tool use sequence proved to be a long-winded affair resulting in a reduction in the effectiveness and speed of the system. For those reasons it was decided to invent new codification as well as index values for tool use sequences. See Appendix E.

6. The index values presented in MOST for the fastening and loosening actions (parameters F&L) proved to be insufficient with regards to the tools utilised. As a result, the index values used in SPAM for fastening operations using tools such as a standard rivet gun were obtained by the allocation of index values of tools that required similar body motions and applied force.

4.6.5 Modules within SPAM

Modules within SPAM refer to sequences generated for particular standard assembly operations namely bolting, screwing and riveting. The process of generating sequences was initially done by utilising set theory. This method provided all permutations given a number of custom parts, standard parts and preferred assembly order. Subsequently, the feasible assembly sequences were extracted form the permutations obtained.

It was found during the validation process that these sequences had to be further narrowed down as some sequences proved to be dependent on the tool used for the operations. For example, the number of feasible sequences for a riveting operation is reduced from six to two when using a standard rivet gun as opposed to an air gun.

Each module has been given a code and a code number. An explanation of how the expressions have been derived is included for each of the standard assembly operations considered.

Each module also has a graphical representation and a corresponding step-by-step description, which allows easy comprehension by the user. This format allows the creation of an assembly workbook or a computer based decision-making system.

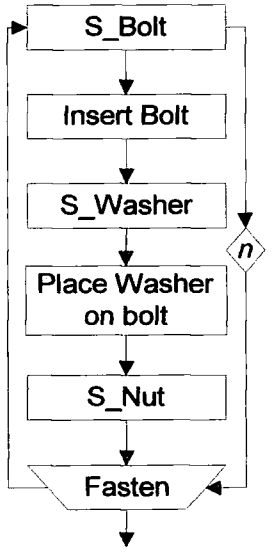
4.6.5.1 Examples of sequences in SPAM

Some examples of assembly sequences derived using SPAM are presented below:

1. Sequences for Bolts, Nuts, and Washers (BNW).

   Reference code: BNW_1 –

   BNW_1 describes a simple bolting operation where all body motions are sequential and there is an absence of complex body motions such as palming actions. A washer is placed on the shank of the bolt before the nut is tightened. The sequence adopted by the assembler is given below. An explanation for the derived equation for sequence BN_1 is given in Table 4-8.



Description of process	**Sequence:**
Collect a handful of bolts form a part bin.	$A_1\ B_0\ G_3$
Insert bolt through holes. Repeat process n times.	$A_1\ B_0\ P_3\ PA_3$
Collect a single washer from a part bin.	$A_1\ B_0\ G_3$
Place washer on bolt	$A_1\ B_0\ P_3$
Collect a single nut from a part bin.	$A_1\ B_0\ G_3$
Fasten with desired tool.	$A_1\ F_3$
Repeat process *n* times.	
Equation:	$23n*\left(2.78^{(-1)}\right)$

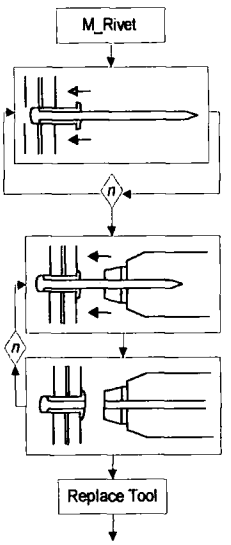
Table 4-8: BNW_1

2. Sequences - Riveting Sequences

   Reference code: RIV_4 –

   RIV_4 describes a riveting operation where all body motions are sequential. RIV_4 is a probable sequence adopted by an assembler riveting a number of parts (e.g. two metal sheets) together. A palming action is introduced when

handling multiple parts. An explanation for the derived equation for sequence BN_1 is given in Table 4-9.



Description of process	Sequence:
Collect a handful of rivets from a part bin.	$A_1\ B_0\ G_3$
Insert rivet into the pre-drilled hole in the material to be joined.	$A_1\ B_0\ P_3$
Repeat n times.	$n\ PA_3$
Insert the rivet head into the nosepiece of the riveting tool and actuate tool.	$I_4\ F_6\ A_1$
Repeat n times.	n
Replace tool.	A_1
Equation:	$(18n+5)*(27.8^1)$

Table 4-9: RIV_4

A full list of all derived standard operation times using SPAM, can be found in Appendix E, together with a guide to reading the diagrams. SPAM.

4.7 Conclusion

The building blocks of CAPABLE*Assembly* have been presented in this chapter. The aggregate product model used to describe the assembled state of a given product; the connectivity model provides a relational product model for the generation of assembly sequences; the assembly time generation algorithm shows how the times attached to each assembly operation have been obtained.

With the requirements for the generation of assembly operations, sequences, and times now in place;

- the **aggregate product model**, used to describe a product in terms of its assembly features and assembly processes

- the **connectivity model**, used to provide relational data between components, and establish assembly constraints for feasibility requirements
- the **assembly time generation algorithm**, used to generate estimate assembly times for assembly operations

the scene is now set for the processes of generating optimal assembly sequences from a product model at the aggregate level of design.

5. Automatic Generation of Optimal Assembly Sequences Using Simulated Annealing

5.1 Introduction

The automatic generation of assembly sequences is recognised as an important aspect of assembly planning and optimisation in order to streamline production and reduce lead times and costs. It also plays an important role in designing and planning the assembly system. The methodology presented herein uses simulated annealing to aid the process of creation and selection of optimal assembly sequences using simulated annealing as a search method. This methodology is centred round an object-oriented platform as a means for managing product, topological, and resource data for assembly modelling and reasoning purposes.

The efficiency of the method is ensured by restricting the search space to assembly sequences that satisfy the connectivity model (discussed in Section 4.3). Since all these sequences already satisfy feasibility criteria, the objective function used for evaluating assembly sequences here is based on criteria that minimise assembly time. The data stored within the connectivity model creates a knowledge base capable of evaluating alternative models of a given product; this feature is necessary if the method is to be of use at the conceptual stage(s) of design.

Generally, assembly sequence generation consists of the two major activities: assembly modelling and generation of feasible assembly sequences. In order to recommend a good sequence of assembly operations for a new product, the process planner needs to be able to select a number of "good" assembly sequences from a pool of feasible options. The process of generating feasible assembly sequences is largely rudimentary and can be performed in an efficient manner when computational methods are employed. However, when considering the evaluation of these sequences the complexity of the process increases exponentially. The use of general-purpose heuristics such as tabu search, local search, simulated annealing and genetic algorithms have been widely acknowledged as an effective method for solving such intractable problems despite the fact that they do not *guarantee* attaining an optimal solution.

This chapter details the multi-criteria optimisation of feasible assembly sequences based on a simulated annealing approach. The layout of this Chapter is as follows:

- Section 5.2 outlines the factors taken into consideration when generating optimal assembly sequences. It also explains the reasoning behind the chosen assembly

criteria for comparing assembly sequences and the method used to compare assembly sequences.

- Section 5.3 details the assumptions made when deriving the objective function used to compare the assembly sequences generated. The assumptions made for generating assembly sequences are also documented.
- Section 5.4 explains the derivation of the objective function that the simulated annealing algorithm uses to compare the assembly sequences. It also outlines the evaluation criteria and mathematical models used to derive the overall assembly time expression.
- Section 5.5 presents a system overview for the assembly sequence optimisation module within CAPABLE*Assembly*. It navigates you through the entire module depicting how the internal modules of the system are interlinked. The following sections discuss the workings of the internal modules in detail.
- Section 5.6 gives an overview of the simulated annealing approach for solving combinatorial optimisation problems, and shows how the method can be adapted to the issue of assembly sequence optimisation.
- Section 5.7 describes the method used to generate an optimal assembly sequence using simulated annealing.
- Section 5.8 presents an illustrative example detailing all the steps mentioned above to generate an optimal assembly sequence for a given product model.
- Finally, Section 5.9 pulls together all the conclusions drawn from this Chapter.

5.2 Definition of Problem

The main objective here is to reduce production lead times and consequently production cost: in other words, to decrease the assembly time of a given product. For a given product consisting of n parts, there are $n!$ possible assembly sequences assuming no technological or geometric constraints. Among the numerous possible assembly sequences, there are x sequences that are actually feasible, that is, would result in the desired functional product. Within this set of feasible assembly sequences there are a series of assembly sequences that optimise the assembly process with respect to a predefined criteria. However, optimisation of the assembly sequence based on one criterion will more than likely be at the expense of another optimisation criterion. Hence, a multi-criterion approach to optimising assembly sequences is required.

There are a number of assembly constraints that can be used to achieve this. For the purpose of this research the following criteria are considered:

- Minimisation of number of reorientations: The effects of re-orientation on the handling time for parts and tools have been documented in Section 2.3.1. This considered only individual parts; here, the focus is shifted to the effects of reorientation on subassemblies.
- Maximisation of stability of intermediate subassemblies: Stability is an important issue when considering assembly time because the presence of unstable subassemblies at best necessitates holding down requirements and increased reorientation problems. At worst, it may cause subassemblies to spontaneously disintegrate.
- Parallelism: An assembly sequence that allows for parallelism can reduce the overall assembly time for a product as it lends itself to the concurrent execution of some assembly tasks.

These criteria were chosen because collectively they significantly affect the assembly time for a given assembly sequence. Clustering (of similar assembly tasks, also referred to as work relatedness) is another typical factor taken into consideration at this stage (Laperriere and ElMaraghy, 1996). If clustering is considered at this stage, the main advantages are:

- Reduction in the handling time for components and their associated tooling requirements.
- Merging of similar operations into simultaneous operations.

However, if clustering is left until the line balancing stage, the assembly time decreases because the operator becomes more skilled at performing the given task. More importantly, restrictions on when certain assembly tasks are performed is relaxed. The key priority is that all operations are loaded on the same workstation and not necessarily at the beginning or end of the assembly sequence, thus increasing the flexibility of the system. In this work we leave clustering considerations till the line balancing stage.

5.3 Assumptions

The automatic generation of assembly sequences is not solely governed by applying assembly constraints. In order to generate all the feasible sequences of assembly operations the following generic assumptions are made:

1) An assembly operation joins two or more parts and/or subassemblies.

2) The order of assembly is the reverse of the disassembly.

3) The geometric relationship between individual parts remains the same after they are assembled.

4) Assembly time is typically regarded as a combination of set-up time and actual assembly time. For the purpose of this research assembly time is considered to be the summation of part handling time and the actual operation time. Tooling/Machine set-up time is not considered. The assumptions made for the generation of the assembly time databases used within the assembly sequence generation module regarding;

    a. handling standard parts (SPAD)

    b. custom assembly parts

    c. standard assembly operations (SPAM)

    d. customised assembly operations

    have been discussed in detail in Chapter4, Section 4.4.

5.4 Optimisation: evaluation criteria and mathematical models

The objective is to develop a mathematical model of the operation sequencing decision process that captures the problems characteristics, and also the assumptions made above. To achieve this, certain control variables or assembly criteria need to be defined. The method approaches the issue of minimising assembly times by considering the following three variables;

- Minimisation of the number of reorientations (c_1)
- Maximisation of parallelism (c_2)
- Maximisation of the stability of the intermediate subassemblies (c_3).

5.4.1. Minimisation of the number of reorientations (c_1)

Some intermediate assembly operations require reorientation of sub-assemblies. The issue of reorientation of components already been accounted for when estimating the handling time for the moving part within an AFC. The objective here is to minimise the number of times sub-assemblies are reoriented while the product is being assembled. To

achieve this we introduce a reorientation index, x_{re}. The cost function for the total number of reorientations is shown in Equation 5-1:

$$RE = \sum_{i=1}^{n-1} x_{re} \qquad \textbf{Equation 5-1}$$

Where,

n is the total number of AFCs for a given assembly sequence.

i is the AFC being considered

The maximum value of RE is equal to $n-1$. Assembly sequences with higher values of RE are said to have higher instances of reorientation.

x_{re} is based on the time standard classification as defined by Boothroyd and Dewhurst (see Appendix A). It is calculated by mapping the assembly time associated with each $\alpha+\beta$ range to a sliding scale of 0 to 1 as shown in Equation 5-2

$$x_{re} = \frac{T_{max} - T_{\alpha+\beta}}{T_{max} - T_{min}} \qquad \textbf{Equation 5-2}$$

Where;

T_{max} is the mean handling time for $\alpha+\beta = 720°$.

T_{min} is the mean handling time for $\alpha+\beta < 360°$.

$T_{\alpha+\beta}$ is the mean handling time for the $\alpha+\beta$ range in question.

If either the moving or stationary part in $AFC_{i,j}$ (notation explained in Section 4.3) is the moving or stationary part in $AFC_{i,j+1}$, then $AFC_{i,j+1}$ is viewed as a component and the net reorientation effect is viewed as the $\alpha+\beta$ value of the moving part in AFC_{ij+1}. Otherwise, if neither AFCs have parts in common, both $AFC_{i,j}$ and $AFC_{i,j+1}$ are viewed as subassemblies and maximum reorientation is imposed on the sequence. This is because typically subassemblies can only be assembled in one direction, this corresponds to an $\alpha+\beta$ of 720°.

x_{re} depends on the value of $\alpha+\beta$ of the moving part in $AFC_{i,j+1}$. The possible values for x_{re} are given below:

1. $\alpha+\beta < 360°$. x_{re} is set to 0.
2. $360° \leq \alpha+\beta < 540°$. x_{re} is set to 0.5.
3. $540° \leq \alpha+\beta < 720$. x_{re} is set to 0.9.

4. $\alpha+\beta = 720$. x_{re} is set to 1.

Take, for example the assembly of the cutting head of a typical hedge trimmer. This consists of the following $AFC_{i,j}$s within the same assembly level (i = 3):

- $Plug'n'Target_{3,9}$
  Mating components: cutting_head_body (stationary) and nut (moving)
  $\alpha+\beta$ values: 540° and 180°
- $Plug'n'Target_{3,8}$
  Mating components: cutting_head_body (stationary) and spacer (moving)
  $\alpha+\beta$ values: 540° and 180°
- $Plug'n'Target_{3,7}$
  Mating components: line_feeder (stationary) and spring (moving)
  $\alpha+\beta$ values: 540° and 180°
- $Placement_{3,6}$
  Mating components: cutting_head_body (stationary) and line_feeder (moving)
  $\alpha+\beta$ values: 540° and 540°
- $Snap\_fit_{3,5}$
  Mating components: cutting_head_body (stationary) and eye (moving)
  $\alpha+\beta$ values: 540° and 720°
- $Placement_{3,4}$
  Mating components: cutting_head_body (stationary) and spool (moving)
  $\alpha+\beta$ values: 540° and 540°
- $Snap\_fit_{3,3}$
  Mating components: cutting_head_body (stationary) and cutting_head_cover (moving)
  $\alpha+\beta$ values: 540° and 540°

The reorientation index (x_{re}) between $Plug'n'Target_{3,9}$ and $Plug'n'Target_{3,8}$ is 0. Since both AFCs have a part in common, in this case the cutting_head_body, $Plug'n'Target_{3,8}$ is viewed as a component. The moving part in $Plug'n'Target_{3,8}$ is the nut with an $\alpha+\beta$ value of 180, hence $x_{re} = 0$.

Using the same analogy, the following x_{re} values and hence RE can be derived for the sequence presented above:

- x_{re} between $Plug'n'Target_{3,8}$ and $Plug'n'Target_{3,7} = 1$.

Both AFCs have no parts in common, hence are both viewed as subassemblies, hence a maximum x_{re} value.

- x_{re} between $Plug'n'Target_{3,7}$ and $Placement_{3,6} = 0.9$.

  Both AFCs have a part in common, $\alpha+\beta$ value for the line_feeder is 540°.

- x_{re} between $Placement_{3,6}$ and $Snap\_fit_{3,5} = 1$.

  Both AFCs have a part in common, $\alpha+\beta$ value for the eye is 720°.

- x_{re} between $Snap\_fit_{3,5}$ and $Placement_{3,4} = 0.9$.

  Both AFCs have a part in common, $\alpha+\beta$ value for the spool is 540°.

- x_{re} between $Placement_{3,4}$ and $Snap\_fit_{3,3} = 0.9$.

  Both AFCs have a part in common, $\alpha+\beta$ value for the cutting_head_cover is 540°.

- $RE = 0+1+0.9+1+0.9+0.9 = 4.7$

5.4.2. Maximisation of parallelism (c_2)

The selection of an assembly plan that allows parallelism leads to significant reduction in the total assembly time. Assembly sequences carried out sequentially generally have longer assembly time when compared to performing operations simultaneously. Economically speaking, this is not directly proportional to a decrease in assembly cost, as simultaneously operations tend to require an increased level of available resources. At the conceptual stage of design, it is advantageous to have some means of measuring the economic trade-off between sequential and concurrent assembly sequences. This assembly variable provides a measure of the ability of an assembly sequence to be carried out simultaneously. The cost associated with parallelism is as defined by Laperrière and ElMaraghy (1992).

Laperrière and ElMaraghy determine how good a disassembly operation is with respect to parallelism by counting the number of components in the two subassemblies, that is the 'child' subassemblies resulting from the disassembly of the 'parent' subassembly. The smaller the difference between the number of components in each child subassembly, the better the operation with respect to parallelism.

If a feasible assembly sequence exists such that every disassembly operation splits the parent subassembly into two subassemblies with equal numbers of components, then maximum parallelism is achieved. If a parent subassembly has an odd number of components, the best that can be achieved is to split this subassembly into two subassemblies whose respective number of components differs by one. If every

operation in the feasible assembly sequence always consists of removing a single component at a time the concurrency cannot exist.

To avoid making a distinction between a parent subassembly with an even or odd number of components, Laperrière and ElMaraghy merge both possible values in the best-case situation are into a single standard, $\overline{dif}$. If "dif" denotes the difference in the component count of the two subassemblies and "n" denotes the number of components in their parent subassembly, then the value of $\overline{dif}$ as defined by Laperrière and ElMaraghy is given by Equation 5-3.

$$\overline{dif} = \begin{Bmatrix} 0 & \text{if diff} = 0 \text{ and n is even} \\ 0 & \text{if diff} = 1 \text{ and n is odd} \\ dif & \text{otherwise} \end{Bmatrix} \qquad \textbf{Equation 5-3}$$

The maximum difference "d" between the component count of two subassemblies resulting from splitting a parent subassembly is given by Equation 5-4:

$$d = n - 2 \qquad \textbf{Equation 5-4}$$

The overall cost function for parallelism PA (Laperrière and ElMaraghy, 1992), is shown in Equation 5-5:

$$PA = \frac{w_{pa} \times \overline{diff}}{d} \qquad \textbf{Equation 5-5}$$

Where w_{pa} is the relative weight of the parallelism criterion as specified by the user, $\overline{dif}$ is the value of the difference in the component count of the two subassemblies as given in Equation 5-3, and d is the maximum value of this difference as given in Equation 5-4.

Assembly sequences with higher values of PA are said to have more instances of parallelism. Take, for example the assembly sequence for a trimmer assembly shown in Table 5-1, its connectivity model is provided in Figure 5-1.

No.	$AFC_{i,j}$	Components within AFC
1	$Wiring_{3,1}$	switch,capacitor
2	$Wiring_{3,2}$	switch,black_wire
3	$Plug'n'Target_{3,7}$	line_feeder,spring
4	$Placement_{3,6}$	cutting_head_body,line_feeder
5	$Snap\_fit_{3,5}$	cutting_head_body,eye
6	$Plug'n'Target_{3,9}$	cutting_head_body,nut
7	$Plug'n'Target_{3,8}$	cutting_head_body,spacer
8	$Placement_{3,4}$	cutting_head_body,spool
9	$Snap\_fit_{3,3}$	cutting_head_body,cutting_head_cover
10	$Placement_{2,1}$	lower_body,motor
11	$Threaded_{2,2}$	cutting_head_ass,motor
12	$Placement_{2,3}$	lower_body,switch_ass
13	$Plug'n'Target_{2,4}$	cable_support,mains_cable
14	$Plug'n'Target_{2,5}$	lower_body,cable_support
15	$Wiring_{2,6}$	switch_ass,mains_cable
16	$Wiring_{2,7}$	motor,switch_ass
17	$Placement_{2,8}$	lower_body,upper_body
18	$Threaded_{2,9}$	upper_body,lower_body,screw

Table 5-1: A feasible assembly sequence for a trimmer assembly

The disassembly of the 'parent' assembly; the trimmer assembly, results in two 'child' assemblies; the switch assembly and the cutting head assembly, as shown in Figure 5-1. If the parent assembly is in Assembly level 1, Assembly level 2 is searched for the presence of subassemblies. In this example there are two subassemblies in assembly level 2, namely the cutting_head_ass and the switch_ass.

The number of AFCs in the cutting_head_ass and the switch_ass is obtained from the connectivity model shown in Figure 5-1. Alternatively, the number of components can be used for the analysis, accessible from the product model also shown in Figure 5-1, but the resulting value of PA remains the same. However, for efficiency and speed, AFCs are used, and the product model is only referenced where absolutely necessary.

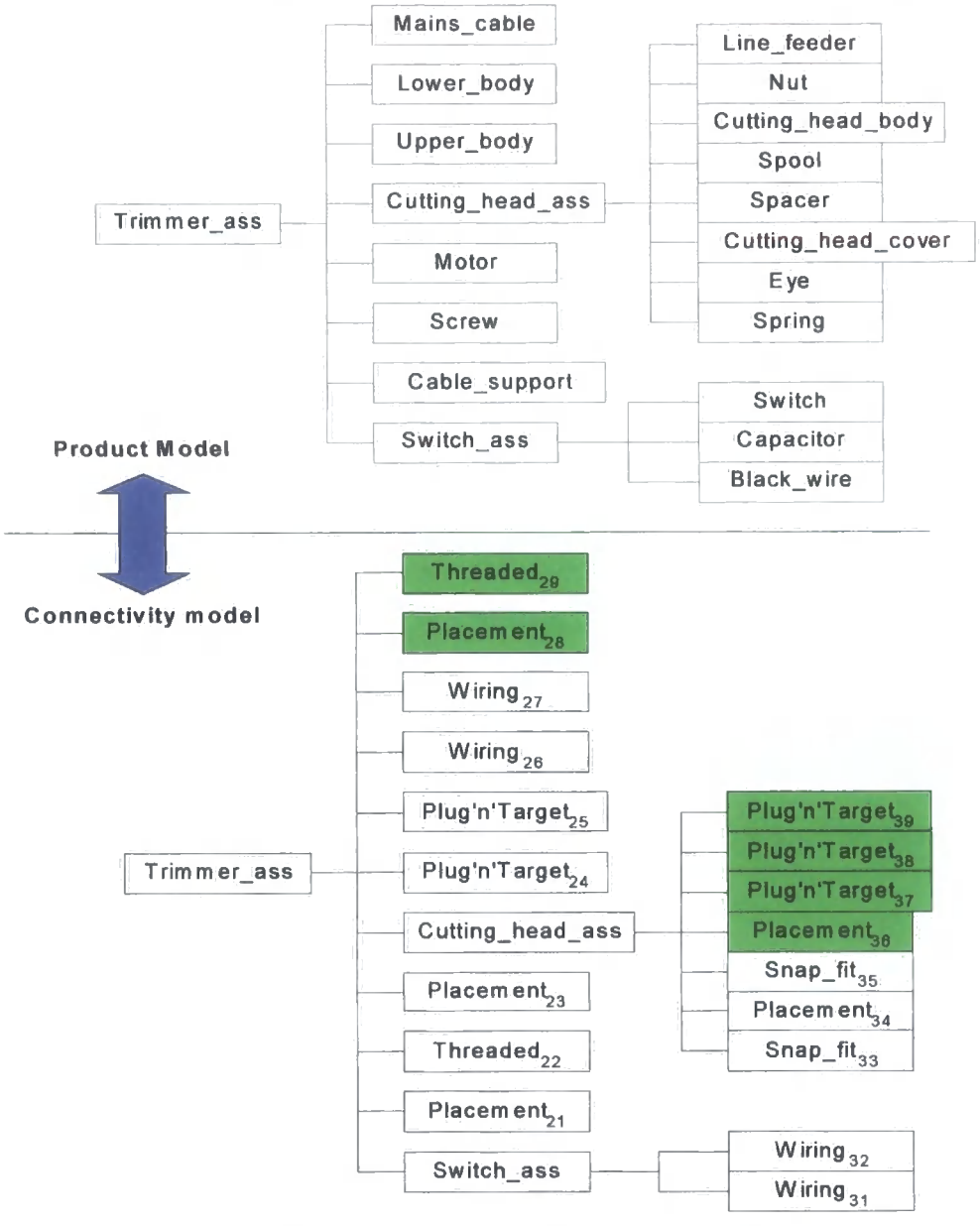


Figure 5-1: Trimmer product model and generated connectivity model

For the assembly sequence in Table 5-1 PA is calculated as follows;

- number of AFCs in cutting_head_ass is 7
- number of AFCs in switch_ass is 2
- $n = 9$. This corresponds to the number of AFCs in the parent assembly, that is, the trimmer_ass.
- from Equation 5-4: $d = 9 - 2 = 7$
- from Equation 5-3: $\overline{dif} = 5$

  AFC difference between cutting_head_ass and switch_ass: $7 - 2 = 5$. Since dif is not equal to 1 or 0, $\overline{dif} = 5$.

- from Equation 5-5: $PA = \frac{w_{pa} \times 5}{7} = 0.7 w_{pa}$

5.4.3. Maximisation of the stability of intermediate subassemblies (c_3)

The aim here is to avoid mating parts that disassemble spontaneously. An assembly sequence that involves highly stable subassemblies increases the ease with which a product/sub-assembly is assembled. Also, the reliability of the sequence of assembly operations is increased resulting in reduced assembly cycle time by preventing errors during execution by avoiding unstable subassemblies when building a product.

One of the basic concepts of good design in terms of DFA is to assemble parts which once in place, are maintained in place by their physical contact with other parts. The stability of an assembly/sub-assembly includes gravity stability, assembly stability and plastic stability.

For the purpose of this research, a stability rating system has been derived, assigning all AFCs (adhesives, plug'n'target, threaded, pressure-fits, riveted, placement, labelling, wiring, packaging and welding) considered with a stability rating index (r_{st}), ranked in ascending order (Appendix G) according to the degree of permanence of the joining methods considered in this research. In general all reversible assembly operations have lower stability ratings, although reversible fastening operations have been assigned high stability rating. A comparison of the joining processes employed can be found in Delchambre (1996).

To compare the stability of two adjacent AFCs ($AFC_{i,j}$ and $AFC_{i,j+1}$) we introduce a stability index, x_{st}. If the two consecutive AFCs do not have mating parts in common, there is no net effect on the stability of the system. Performing $AFC_{i,j}$ before $AFC_{i,j+1}$, is as likely to stabilise the assembly as destabilise the assembly. As such no significant information can be inferred from the stability index as the action $AFC_{i,j+1}$ does not affect any of the parts in $AFC_{i,j}$. Hence, x_{st} is set to zero.

If the two adjacent AFCs have mating parts in common, a comparison is made between their stability-rating indices.

- If the stability of $AFC_{i,j}$ is greater than $AFC_{i,j+1}$; x_{st} is set to –1, system is potentially unstable. If the following assembly operation to a given operation does not secure the relative position of one of the parts in the preceding operation, the stability of the assembly is questionable. The mere fact the system

could disassemble or require holding down is sufficient to impose a negative stability index.

- If the stability of $AFC_{i,j}$ is less than $AFC_{i,j+1}$; x_{st} is set to 1, system is potentially stable. If the following assembly operation to a given operation secures one of the parts in the preceding operation, the stability of the system is not guaranteed, but the assembly might be stable. Again, this is sufficient to impose a positive stability index.
- If the stability of $AFC_{i,j}$ is equal to $AFC_{i,j+1}$; x_{st} is set to 0. Since no net effect can be inferred from the stability rating, no statement is made with regards to stability.

The logic described above is given in Equation 5-6.

$$x_{st} = \begin{cases} -1 & AFC_{i,j}(r_{st}) > AFC_{i,j+1}(r_{st}) : \text{mating part in common} \\ 0 & AFC_{i,j}(r_{st}) = AFC_{i,j+1}(r_{st}) : \text{mating part in common} \\ 1 & AFC_{i,j}(r_{st}) < AFC_{i,j+1}(r_{st}) : \text{mating part in common} \\ 0 & \text{no mating part in common} \end{cases} \quad \textbf{Equation 5-6}$$

The cost function for the maximisation of stability is:

$$ST = \sum_{i=1}^{n-1} x_{st} \quad \textbf{Equation 5-7}$$

Where,

n is the total number of AFCs

i is the AFC being considered

Assembly sequences with high values of ST are said to have more instances of stability. The maximum value of ST is equal to n-1.

Take, for example the assembly sequence presented in Section 5.4.1

The stability index (x_{st}) between $Plug'n'Target_{3,9}$ and $Plug'n'Target_{3,8}$ is 0. Since both AFCs have the same stability rating index ($r_{st} = 5$).

Using the same analogy, the following x_{st} values and hence ST can be derived for the sequence presented in Section 5.4.1:

- x_{st} between $Plug'n'Target_{3,8}$ and $Plug'n'Target_{3,7} = 0$.
  AFCs have the same stability-rating index. ($r_{st} = 5$)
- x_{st} between $Plug'n'Target_{3,7}$ and $Placement_{3,6} = -1$.

AFCs have a part in common, Placement$_{3,6}$ (r_{st} = 3) occurs after Plug'n'Target$_{3,7}$ (r_{st} = 5)

- x_{st} between Placement$_{3,6}$ and Snap_fit$_{3,5}$ = 1.
  AFCs have a part in common, Snap_fit$_{3,5}$ (r_{st} = 6) occurs after Placement$_{3,6}$ (r_{st} = 3)
- x_{st} between Snap_fit$_{3,5}$ and Placement$_{3,4}$ = -1
  AFCs have a part in common, Placement$_{3,4}$ (r_{st} = 3) occurs after Snap_fit$_{3,5}$ (r_{st} = 6)
- x_{st} between Placement$_{3,4}$ and Snap_fit$_{3,3}$ = 1
  AFCs have a part in common, Snap_fit$_{3,3}$ (r_{st} = 6) occurs after Placement$_{3,4}$ (r_{st} = 3)
- $ST = 0+0+(-1)+1+(-1)+1 = 0$

5.4.4. Minimisation of assembly time; An Overall expression

An expression for the minimisation of assembly time can be defined by an overall assembly variable (c). Since the overall operation time for each AFC within an assembly sequence is independent of the assembly sequence, the overall expression for the minimisation of assembly time can simply be derived by normalising the assembly variables c_1, c_2, c_3 (see Section 5.4) for each assembly sequence generated, and by applying weighting factors w_{re}, w_{pa}, w_{st} respectively, see Equation 5-8.

- c_1 = RE
- c_2 = PA
- c_3 = ST

This allows the user to define the relative priorities for analysis. Assembly sequences with the higher values of c denote near optimal assembly sequences, with the highest stored value indicating the most favourable assembly sequence.

$$c = (w_{re})\frac{X_{re}}{c_1} + (w_{pa})^* c_2 + (w_{st})\frac{c_3}{X_{st}} \qquad \textbf{Equation 5-8}$$

Where;

- X_{re}, maximum possible value of reorientation index used to normalise c_1
- X_{st}, maximum possible value of stability index used to normalise c_2

The optimisation process is controlled by the following assembly variables: (*i*) parallelism; (*ii*) number of re-orientations; (*iii*) stability of subassemblies; (iv) 'globally' good sequences; and (*v*) 'locally' good assembly sequences:

- Globally good optimisation, if an assembly plan is globally good, then all assembly variables described in Section 5.5 are considered.
- Locally good optimisations, if an assembly plan is locally good, then at least one of the assembly variables are described in Section 5.5 has been selected. Assembly variables can be selected or deselected by assigning their respective weighting factor to zero.

It is important to remember the validity of an assembly sequence is different from the feasibility of an assembly sequence. An assembly sequence is feasible if it results in the functional end product, as designed. The validity of an assembly sequence is subjective, hence the provision of locally and globally optimised assembly sequences.

5.5 Proposed method for assembly sequence generation

5.5.1. System overview

The module takes as its input the connectivity model as shown in Figure 5-1. The initial assembly sequence generated from the connectivity model (see Section 5.7.1) is the feasible assembly sequence used as the input to the simulated annealing algorithm. The assembly sequence is encoded using random numbers (Bean, 1994) to supply the simulated annealing algorithm with an initial solution.

Within the simulated annealing process, an assembly sequence is generated by randomly interchanging neighbouring AFCs. The sequences generated are decoded and evaluated using the overall expression for minimising assembly time (Equation 5-8). The process is repeated locally until the best solution is attained based on its objective function value. Once the termination criterion has been attained (see Section 5.7.3), the resulting assembly sequence is the best possible assembly sequence. It is likely the optimal assembly sequence is the same as the initial assembly sequence. The overall configuration of the assembly sequence optimisation module within CAPABLE*Assembly* is shown in Figure 5-1. The output of the module is an optimised assembly sequence.

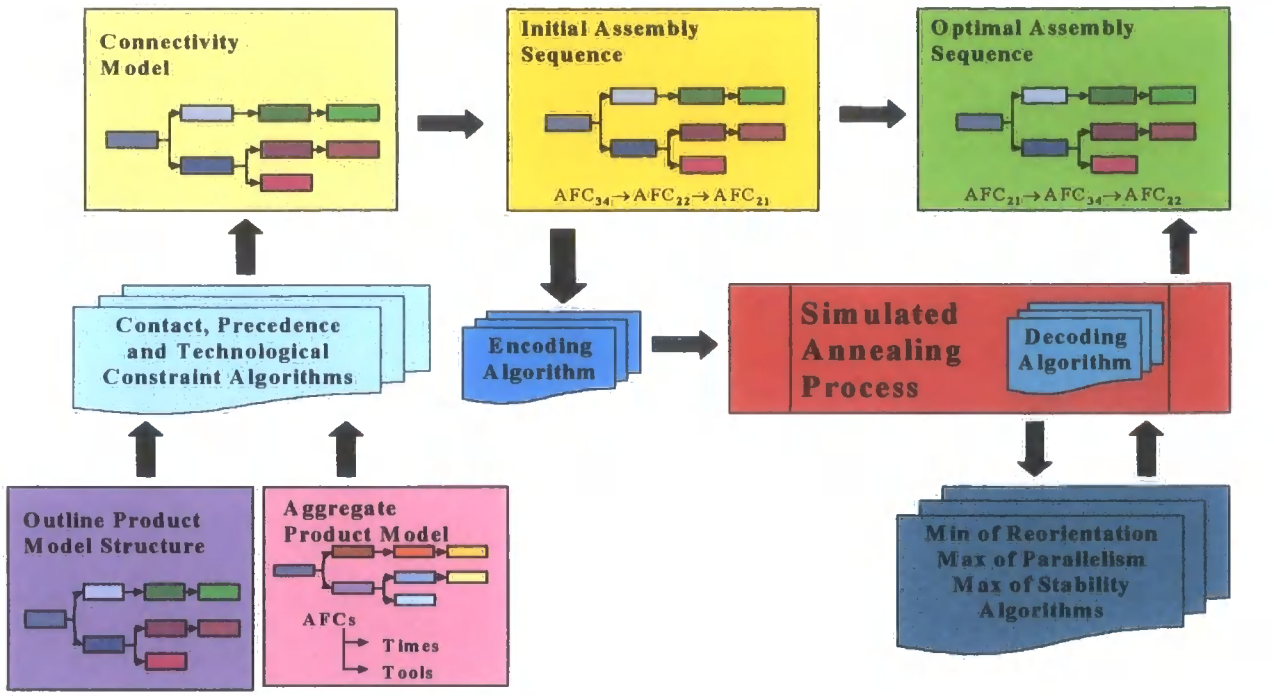


Figure 5-2: Overall module structure

5.6 *Simulated Annealing (SA) Algorithm*

SA algorithms have been used successfully in solving various combinatorial optimisation problems, including VLSI design (Gerez, 1999), scheduling (Kim and Kim, 1996), and assembly line balancing (Suresh and Sahu, 1994). The SA approach can be viewed as an enhanced version of local optimisation or an iterative improvement, in which an initial solution is repeatedly improved by making small local alterations until no such alteration yields a better solution. It has already been stated that the use of simulated annealing for sequence optimisation does not guarantee an optimal solution. However, they generally provide a good solution and statistically guarantee finding a close to the best possible solution.

KirkPatrick et al (1983) first developed the simulated annealing (SA) method. Simulated annealing emulates the annealing process in thermodynamics. The essence of the method is slow cooling to allow ample time for redistribution of energy; hence the temperature of the system is the controlling factor. In the process concerned, a solid material is first heated up to a temperature that allows all its molecules to move freely (solid becomes liquid) and then is cooled very slowly until it crystallises with a perfect lattice. The rate of cooling determines the lattice. At the end of the process, the total energy of the material is minimal provided that the cooling is very slow.

According to Kirkpatrick et al, simulated annealing is based on the Metropolis procedure developed in the field of statistical mechanics. The Metropolis method

combines iterative improvement with controlled uphill moves in search of a global optimal solution. The uphill move allows the procedure to escape local optima. A comparison of simulated annealing with other algorithms (Johnson et al, 1989) demonstrated that the technique performed better than other algorithms in many cases.

The basic approach of simulated annealing is to search for the optimal solution by randomly perturbing the system from its current state to a neighbouring state. Neighbouring states are loosely defined as those states that can be reached by random perturbations from a given state. The perturbations must be made in such a way that the states generated remain within the neighbourhood long enough to discover local optima that exist within the neighbourhood. The state solution value determines whether a solution should be accepted as a new solution or be rejected in favour of the previous solution. When a transition is made to a new state, it is evaluated and becomes the current state if it has a lower state solution value than the best state found previously. The method in which state perturbations are allowed to occur and the length of time the states are perturbed within the neighbourhood control the likelihood of a state with a higher value becoming the current state.

Starting from an initial solution (assembly sequence) f, the SA generates a new solution (new assembly sequence) g in the neighbourhood of the original solution f. The change in the objective function value, $\Delta c = c(f) - f(g)$, is calculated, where c, is the objective function. The objective function used for the purpose of this research is the overall assembly variable given in Equation 5-8. For a minimisation problem, if $\Delta c < 0$, the transition to the new solution is accepted according to the negative probability distribution expressed in Equation 5-9.

$$e^{\frac{-\Delta c}{T}}$$ **Equation 5-9**

Where,

c is the objective function to determine the state value; the overall assembly variable

T is the control parameter; temperature

SA algorithms generally start at a higher temperature; at each temperature a search is carried out on the local space for pre-specified number of iterations, that is, the epoch length. The temperature is gradually lowered according to a given cooling rate. Higher values of cooling rate correspond to a slower cooling process allowing the exploration on a larger search space and thus preventing premature convergence providing a false

optimised assembly sequence. The analogy with the physical model has the following points of correspondence with the issue of assembly sequence generation.

- The energy corresponds to the objective function used to evaluate the state value. This research uses the overall assembly variable, c, as expressed in Equation 5-8 as its objective function.
- The movement of the molecules correspond to the sequence of the moves in a set feasible assembly sequences.
- The temperature corresponds to a control parameter T which controls the acceptance probability for a move from $f \in F$ to $g \in N(f)$.

5.7 Assembly sequence generation; Simulated Annealing

As described in Chapter 4, simple heuristics are applied to determine base and moving parts in each AFC and the pre-determined ranking of the AFCs within assembly levels coupled with a simple bottom-top assembly approach results in an initial assembly sequence. A browser was designed to check the details of the generated assembly sequence as shown in Figure 5-3.

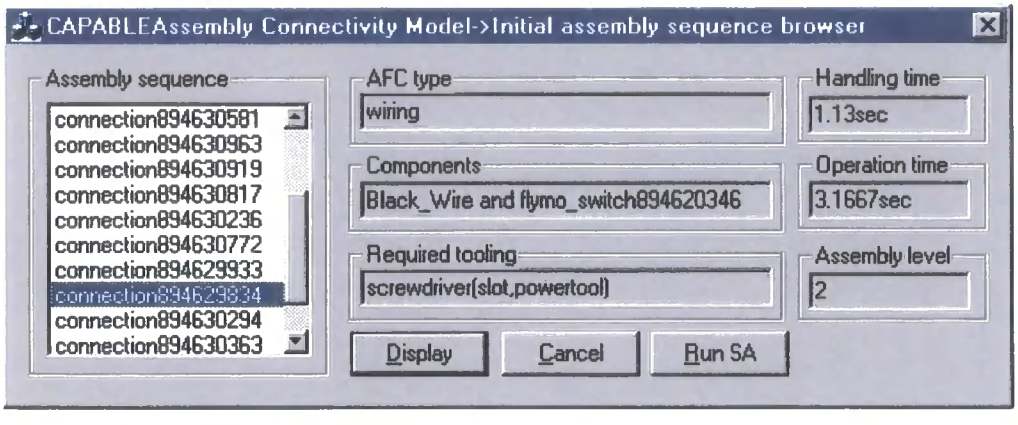


Figure 5-3: Initial assembly sequence

This was done to ensure all the details required to ascertain the properties stored within each AFC of an assembly sequence was sufficient to access the sequence according to the assembly criteria listed in Section 5.4.

There are various methods used for representing solutions in sequencing and optimisation problems when solving heuristic problems. The major difficulty is in ensuring the feasibility of the generated solutions. Whilst this is a problem for other heuristic methods, the issue is of little importance in the case of simulated annealing. This is mainly due to the fact the process of generating a new solution is done by interchanging two random components. In general, this should reduce the feasibility

problem as only slight changes are made at a given time to ensure the new 'state' is within the neighbourhood of the current state. Here, AFCs are used as opposed to components, and the feasibility problem is further reduced.

As already stated, the initial sequence generated from the connectivity model is a feasible sequence as are all sequences generated from the connectivity model. One approach to maintain feasibility would be to generate the new solution from the connectivity model, and subsequently encode the solution. Although this would easily guarantee feasibility, the process is time consuming, and it does not guarantee the new solution to be within the neighbourhood of the original solution thus the local optimisation is redundant. To maintain feasibility, the fixed and floating AFC are used to limit the random metamorphosis of the assembly sequences generated

It was found necessary to limit the random movement of AFCs to floating AFCs. The rules that govern the generation of new sequence were derived through a series of trial and error runs, gradually increasing the restrictions until almost every assembly sequence checked using the assembly browser shown in Figure 5-3 remained within the feasibility framework. These restrictions include:

1. Only two floating AFCs can be randomly chosen and interchanged to create a new solution at any given time. This helps to maintain the new solution within the neighbourhood of the current solution. It was found that freedom for all AFCs is not conducive for feasibility.

2. The movement of floating AFCs are further restricted to a top-down motion. That is, AFCs in level two can be moved to assembly level 3, but the reverse is not allowed. This effectively limits the random interchanging process to within assembly levels. If two AFCs are chosen such that the reverse movement is required, the AFCs in the higher assembly level is moved and the other AFC remains in its original position. Higher assembly level refers to a higher assembled state; AFCs in lower numbered assembly levels are regarded to be of a higher assembly level. For example, if $AFC_{2,3}$ is chosen to interchange with $AFC_{3,2}$, $AFC_{3,2}$ retains its position and $AFC_{2,3}$ is placed immediately before $AFC_{3,2}$.

3. Fixed AFCs are fixed with respect to their assembly level, and their neighbouring AFCs.

5.7.1. Sequence representation: Encoding

For the purpose of the research it is of great importance the encoding method possess an inherent simplicity in terms of decoding, as this will drastically affect the computational time due to the number of evaluation criteria being considered. When encoding the solution, the concept of random keys as suggested by Bean (1994) has been adopted.

The random keys representation encodes a solution with random numbers. The assembly sequence is encoded into a string of randomly generated numbers ranging between 0 and 1, the magnitude of these numbers representing the position of the AFCs within a generated assembly sequence. In the encoding scheme, values of numbers in the string are originally selected from uniform random numbers in the region [0,1]. These values are used as a sort keys to decode the solution. The primary difference between this encoding and those in the literature is the use of random numbers as tags to represent solutions.

For example, consider the following assembly sequence containing eight floating AFCs, within the same assembly level:

- $\text{Placement}_{2,1}$
- $\text{Threaded}_{2,2}$
- $\text{Placement}_{2,3}$
- $\text{Plug'n'target}_{2,4}$
- $\text{Plug'n'target}_{2,5}$
- $\text{Wiring}_{2,6}$
- $\text{Wiring}_{2,7}$
- $\text{Placement}_{2,8}$

Eight real-valued random numbers between 0 and 1 are generated using a standard random number generator, $Rand(x)$ to represent the sequence

- $\text{Placement}_{2,1} \equiv 0.41$
- $\text{Threaded}_{2,2} \equiv 0.18$
- $\text{Placement}_{2,3} \equiv 0.63$
- $\text{Plug'n'target}_{2,4} \equiv 0.26$
- $\text{Plug'n'target}_{2,5} \equiv 0.19$

- $Wiring_{2,6} \equiv 0.15$
- $Wiring_{2,7} \equiv 0.11$
- $Placement_{2,8} \equiv 0.29$

Using the mapping: 0.41**(1)**, 0.18**(2)**, 0.63**(3)**, 0.26**(4)**, 0.19**(5)**, 0.15**(6)**, 0.11**(7)**, 0.29**(8)**

This would yield the sequence: 7 → 6 → 2 → 5 → 4 → 8 → 1 → 3, which is easily decoded to the assembly sequence:

- $Wiring_{2,7} \equiv 0.11$
- $Wiring_{2,6} \equiv 0.15$
- $Threaded_{2,2} \equiv 0.18$
- $Plug'n'target_{2,5} \equiv 0.19$
- $Plug'n'target_{2,4} \equiv 0.26$
- $Placement_{2,8} \equiv 0.29$
- $Placement_{2,1} \equiv 0.41$
- $Placement_{2,3} \equiv 0.63$

5.7.2. Simulated Annealing Parameters

The basic parameters of the simulated annealing assembly sequence algorithm are as follows:

1. Initial temperature *T*; for the purpose of this research a typical value of 100 is used.
2. A maximum number of cooling schedules (*Mcs*) = 5 has been chosen for the purpose of this study. The cooling schedules represent a finite time implementation for the simulated annealing algorithm. *cs* is used to represent a single cooling schedule.
3. The random number *R* (between 0 and 1), used for probability acceptance within the SA algorithm is generated using a standard random number generator *Rand*(*x*). The standard number generator function is 'wrapped' to ensure an even distribution of the numbers generated. *R* = *Random* (*x*).
4. Interchanging two random floating AFCs generates the next assembly sequence.

5. The cooling rate *cr*, is the rate of change of temperature with increasing number of cooling schedules. A cooling rate of 0.95 is used. This means that the cooling rate is low to allow for adequate exploration of a particular search space. This value was established by trail and error. However, it is widely acknowledged that a value above 0.9 will yield reasonable results.

6. At a particular temperature, if the ratio of accepted solutions to number of solutions generated is less than a predetermined number, the cooling schedule is increased by 1. A value of 0.1 has been adopted. This is to say the iterative process does not leave the locality until the majority of new sequences generated are being rejected.

7. *NSol*, is the number of new solutions generated.

8. *TSol*, is the total number of solutions accepted.

A flow chart showing the SA process presented is shown in Figure 5-4.

5.7.3. Pseudo-code for assembly generation sequence

The algorithm itself consists of an outer loop in which the temperature is gradually lowered and an inner loop in which the assembly sequence is randomly altered by interchanging AFCs within the assembly sequence. These newly generated assembly sequences are either accepted or rejected based on their objective function value. The inner loop is executed until the majority of solutions generated are largely being rejected, at this stage thermal equilibrium is attained and the code returns to the outer loop and the process is repeated. See pseudo code in Figure 5-5.

The strategy for accepting or rejecting assembly sequences is represented by the function 'accept' in the pseudo code. Within the accept function, the function '*Rand*(*k*)' generates a real-valued number between 0 and k, with a uniform distribution. Here, k is equal to one. The function 'new temperature' computes the new lower temperature to be used for the next execution of the inner loop. The function 'stop' finally decides whether to terminate the search.

As simulated annealing is prone to visiting the optimal solution and moving away from it, the best solution is kept as a separate variable and is reported back at the end of the search instead of the final value assembly sequence stored. The combination of the thermal-equilibrium, new-temperature, and stop functions define what is known as the cooling schedule (cs). Theoretical analysis shows that the cooling schedule can be chosen in such a way that the probability of finding the global optimum is becomes

equal to one. Hajek (1988) and Van Laarhoven and Aarts (1987) discuss the conditions necessary for convergence of the simulated annealing algorithm.

However, these schedules imply a very large number of moves before stopping. As the primary aim is to obtain a good solution as fast as possible, a cooling schedule that does not guarantee an optimal is used. A value of 5 was determined using a trial and error method based on five different products consisting of varying number of AFCs. In general, it was found that a value of five with slightly higher value for cooling rate was sufficient to attain a near optimal solution.

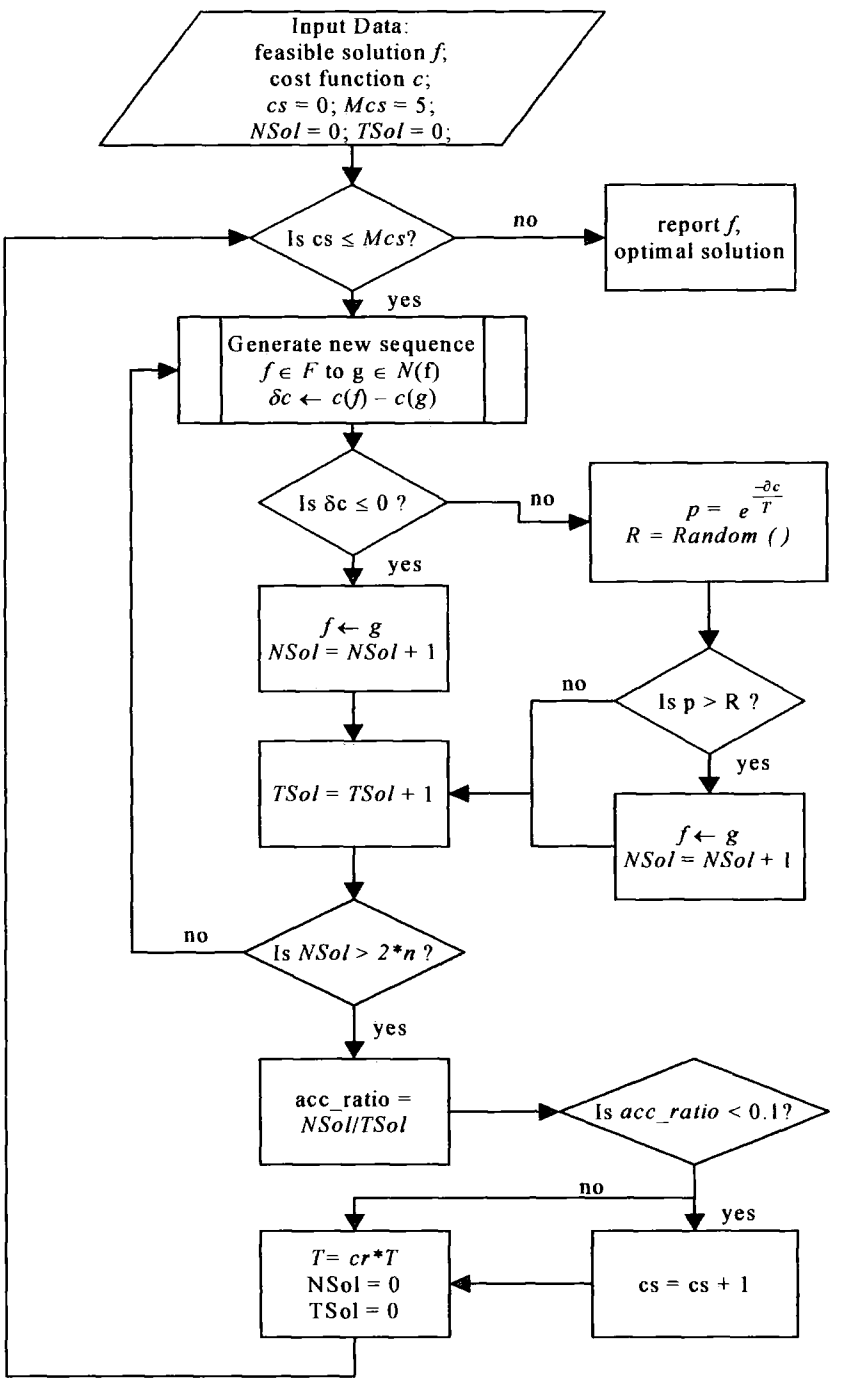


Figure 5-4: Flowchart for simulated annealing algorithm for sequence optimisation

The code used was written in C++. A pseudo code for the generation of an optimal assembly sequence is provided in Figure 5-5.

```
int accept(struct feasible_solution f, g)
{
        float δc;
        δc ← c(f) − c(g);
        if (δc ≤ 0)
                return 1;
        else return ( e^(−δc/T) ) > random(1));
}
int stop(cooling_schedule cs, max_cooling_schedule Mcs,
no_of_accepted_solutions NSol, no_of_total_solutions TSol)
{
        int cs;
        int Mcs;
        int Nsol;
        int Tsol;
cs ← cs + 1;
        if (cs > Mcs);
                return 1;
        else
                T ← new_temperature(T);
                NSol = 0;
                Tsol = 0;
                return 0;
}
int thermal_equilibrium(int Nsol, no_of_components n)
{
        if (Nsol > 2*n)
                return 1;
        else return 0;
}
simulated annealing( )
{
        struct feasible_solution f, g;
        float temperature T;
        f ← initial solution ();
                do {
                        do {
                                g ← "some element of N(f), new solution";
                                        if (accept(f, g))
                                                f ← g;
                                                NSol = NSol + 1;
                                                TSol = TSol +1;
                        while (!thermal_equilibrum( ));
                        T ← new_temperature (T);
                        NSol = 0;
                        TSol = 0;
                while (!stop);
"report f";
```

Figure 5-5: Pseudo-code for generation of optimal assembly sequence

5.8 Illustrative Example

The performance of the method presented was tested on a number of products. The example presented uses a lightweight outdoor product. A simplified product model and the corresponding connectivity model are shown in Figure 5-1. The contact, precedence and technological constraints algorithms were employed to generate the corresponding connectivity model from the aggregate product model. Thereafter, a simple top down assembly sequence generator based on the successive determination of moving parts, base parts, part weight and part size is applied to generate an initial assembly sequence which is in turn used as the initial solution for the SA algorithm.

The AFCs shaded green in Figure 5-1 are fixed AFCs, all other AFCs are regarded to be floating AFCs. Movement of AFCs are limited to a top down motion, that is, AFCs in level 2 can move to level 3 but the reverse is not permitted. Table 5-2 shows the sequence of AFCs generated from the top-down sequence generator; the mating components and the encoded strings are also shown in Table 5-2. This is the initial solution (initial assembly sequence) fed into the simulated annealing program.

Two optimisation procedures were performed; a locally good optimisation was performed prior to a globally good optimisation. A locally good optimisation is achieved by applying suitable weightings to the assembly variables to effectively ignore certain assembly variables. For example, if reorientation is to be ignored, as it was in this case, the weighting factor for reorientation in Equation 5-4 should be $w_{re} = 0$. Table 5-2 shows a locally good optimisation with; $w_{st} = 0.5$, $w_{re} = 0.0$, $w_{pa} = 0.5$. The numbers chosen, as weights are arbitrary and only serve to amplify the relative magnitude of an assembly variable.

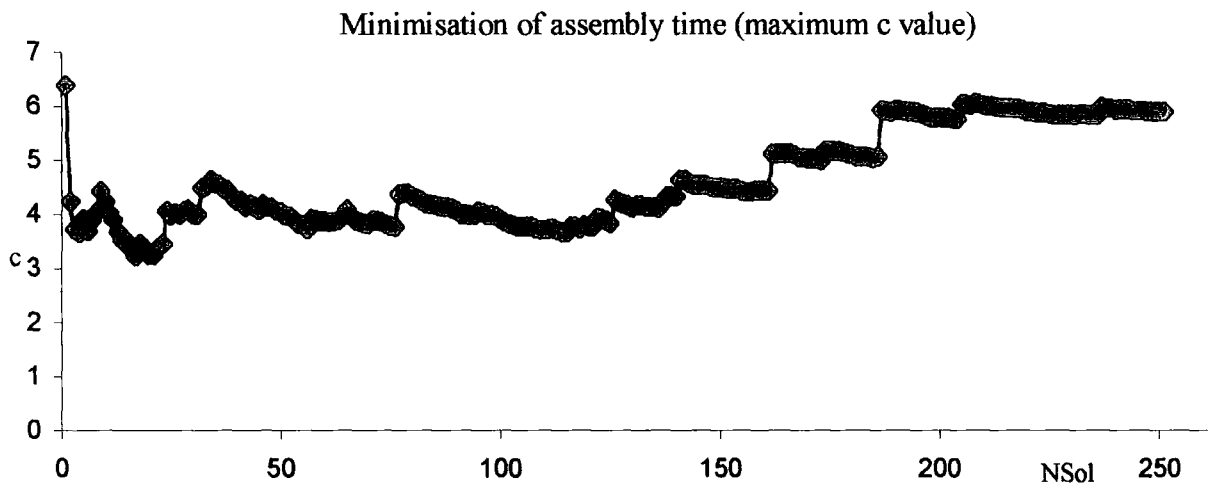


Figure 5-6: Variation of overall assembly variable

In the case of a globally good optimisation, all three assembly variables are considered. Table 5-3 shows a globally good optimisation with; $w_{st} = 0.5$, $w_{re} = 0.5$, $w_{pa} = 0.5$. Figure 5-6 shows the assembly variables of each assembly sequence generated in the 5th cooling schedule. The analysis used 5 cooling schedules. As can be seen from Figure 5-6, the 5th cooling schedule starts with a good solution shown by the high value of c, (scale has been modified for graphical purposes). The solution then deviates, the overall assembly variable starts to increase, and gradually the annealing process begins to finds better solutions as the temperature is slowly decreased. Initially the solutions are clustered a false local optimum is reached, as the temperature is decreased further, the pattern evens out and a true optimal solution is found. The process can be speeded up, if

one uses the number of solutions found as the stopping criteria. As can be seen from Figure 5-6, the main problem with this is you do not tend to catch false local optimal solutions. At the moment, the system stops when it continuously accepts almost all the new sequences generated. This method of convergence generally tends to yield better solutions. Indeed, if the number of cooling schedules were to be increased, it might yet yield a better solution. However, when balanced against computational time, five to six cooling schedules was found to be more than sufficient for all the case studies considered (maximum CPU of 60).

The globally good optimisation process introduced the reorientation criterion. The general trend of the assembly sequence generated follows that of the locally good solution, with AFCs migrating from lower levels to higher assembly levels, increasing stability and parallelism. The effect of introducing the reorientation index is evident by the new sequence generated for the assembly of the cutting head. The system also suggests assembling the "cutting head" assembly prior to the switch assembly. The optimisation processes have created what appears to be a completely top-down assembly direction for this subassembly. This has satisfied the reorientation criterion.

The assembly sequences generated for both locally and globally optimised scenarios were mapped to a predefined assembly line layout to establish assembly times for the assembly plans generated. The estimated difference in total assembly time between the locally and globally optimised scenarios is approximately 8%, with the global solution offering the lowest time.

Both locally and globally good optimisation routines produced good results. In the case of locally good optimisation, it can be seen from Table 5-2 that several AFCs have moved from level 2 to level three. Notably, the mains_cable and cable_support are now members of the switch assembly. Also, the motor has been included to the cutting_head_ass, thus satisfying the parallelism criterion. The stability of the assembly sequence was also increased as a result of the optimisation process. Post-optimisation, the switch assembly is assembled and placed on the base part (that is, secured under the influence of gravity and vibration) before other parts are attached to the switch assembly.

No.	AFCs (afrtype)	Components within AFC
1	$Wiring_{31}$	switch,capacitor
2	$Wiring_{32}$	switch,black_wire
3	$Plug'n'Target_{24}$	cable_support,mains_cable
4	$Plug'n'Target_{25}$	lower_body,cable_support
5	$Placement_{23}$	lower_body,switch_ass
6	$Wiring_{26}$	Switch_ass,mains_cable
7	$Plug'n'Target_{37}$	Line_feeder,spring
8	$Placement_{36}$	cutting_head_body,line_feeder
9	$Snap\_fit_{35}$	cutting_head_body,eye
10	$Plug'n'Target_{39}$	cutting_head_body,nut
11	$Plug'n'Target_{38}$	cutting_head_body,spacer
12	$Placement_{34}$	cutting_head_body,spool
13	$Snap\_fit_{33}$	cutting_head_body,cutting_head_cover
14	$Threaded_{22}$	cutting_head_ass,motor
15	$Placement_{21}$	lower_body,motor
16	$Wiring_{27}$	motor,switch_ass
17	$Placement_{28}$	lower_body,upper_body
18	$Threaded_{29}$	upper_body,lower_body,screw

Table 5-2: Results of locally good analysis

No.	AFCs (afrtype)	Components within AFC
1	$Plug'n'Target_{39}$	cutting_head_body,nut
2	$Plug'n'Target_{38}$	cutting_head_body,spacer
3	$Plug'n'Target_{37}$	Line_feeder,spring
4	$Placement_{36}$	cutting_head_body,line_feeder
5	$Snap\_fit_{35}$	cutting_head_body,eye
6	$Placement_{34}$	cutting_head_body,spool
7	$Snap\_fit_{33}$	cutting_head_body,cutting_head_cover
8	$Threaded_{22}$	cutting_head_ass,motor
9	$Placement_{21}$	lower_body,motor
10	$Wiring_{31}$	switch,capacitor
11	$Wiring_{32}$	switch,black_wire
12	$Plug'n'Target_{24}$	cable_support,mains_cable
13	$Plug'n'Target_{25}$	lower_body,cable_support
14	$Placement_{23}$	Lower_body,switch_ass
15	$Wiring_{26}$	Switch_ass,mains_cable
16	$Wiring_{27}$	motor,switch_ass
17	$Placement_{28}$	lower_body,upper_body
18	$Threaded_{29}$	upper_body,lower_body,screw

Table 5-3: Results of globally good analysis

5.9 Conclusions

The main purpose of this research is to create a system suitable for the automatic generation of an optimal assembly sequence for a given product at the early stages of design. The method is based on the creation of an aggregate product model and the subsequent extraction of contact, precedence and technological relationships from the aggregate product model to create a connectivity model. The extraction of such relationships facilitates the generation of an initial rudimentary assembly plan, which

reduces the search space, as it is a well-known fact that the simulated annealing process benefits from a good initial solution. The generated assembly plan is then refined through a series of optimisation methods using simulated annealing. The simulated annealing algorithm seeks to optimise an assembly rating variable, which includes functions for reorientation, parallelism and stability.

The ability of the system developed to quickly generate and optimise assembly plans locally as well as globally when various criteria are enabled or their relative importance is changed makes it an effective tool for simultaneously considering several manufacturing considerations at the design stage. The results obtained using the method presented have been very encouraging. Indeed, a total of four industrial products (presented in Chapter 7) have been modelled and optimal sequences generated and implemented, with good results.

It is important to note that the generation of an optimal assembly sequence does not in itself imply an optimal assembly plan. An optimal assembly plan can only be realised when the available resources, human and equipment, are taken into consideration; assembly line balancing.

6. Balancing Single-Model Assembly Lines (SALB): A genetic algorithm approach

6.1 Introduction

Ideally, a design/production engineer would prefer to be presented with a number of 'good' assembly plans that take into consideration available factory and resource constraints and assembly line balancing. This chapter presents a computer-based methodology for the Single-Model Assembly Line Balancing (SALB) problem using Genetic Algorithms (GAs). It involves assigning individual assembly operations to workstations such that certain constraints are satisfied and some specified objective achieved. This method aims at generating a number of optimal solutions that lead to maximum production rate, workload smoothness, work-relatedness, and worker allocation.

The SALB problem can be seen as the minimisation of number of workstations and cycle time for a given set of partially ordered assembly operations (Scholl, 1995; Johnson, 1998; Hoffman, 1990; Hackman, 1989). It is only recently that the overall 'goodness' of an assembly line has been taken into consideration. Typical measurements of the goodness of an assembly line, include workload smoothness, work-relatedness, and optimised assembly sequences. Although these factors are now being discussed, they are seldom optimised in a holistic manner. One of the more holistic approaches to solving the SALB problem is offered by Ma (1997), where the best solution is obtained by maximising production rate and minimising workload variance. Kim, Kim, and Kim (1998) look at workload smoothing on assembly lines, and further extend their methodology to the issue of balancing two-sided assembly lines (Kim, Kim, and Kim, 2000). Amen (2000) present an exact method for cost-oriented assembly line balancing based on minimising the number of workstations.

This chapter presents an algorithm for the generation of near optimal assembly plans for a given set of optimised assembly sequences based on a series of performance measures, namely, workload smoothness, work-relatedness (taking into consideration precedence of loaded assembly sequence), worker allocation, and, where deemed necessary, an estimated assembly cost. The method aims to carefully balance all performance measures considered, by using objectives such as minimisation of number of workstations and cycle time as initialisation parameters.

The layout of this Chapter is as follows:

- Section 6.2 outlines the factors taken into consideration when solving SALB problems. It also explains the reasoning behind the chosen performance criteria when evaluating the goodness of an assembly plan.
- Section 6.3 details the assumptions made for balancing assembly lines. This includes the assumptions made for the derivation of the objective functions for minimisation of workstations and minimisation of cycle time for a given assembly plan. Assumptions with regards to the factor model are also detailed.
- Section 6.4 outlines the evaluation criteria and mathematical models used to measure the performance of assembly lines. It also explains the derivation of the objective function the genetic algorithm uses to compare the assembly plans.
- Section 6.5 presents a system overview for the assembly planning optimisation module within CAPABLE*Assembly*. It navigates you through the entire module depicting how the internal modules of the system are interlinked. The following sections discuss the workings of the internal modules in detail.
- Section 6.6 describes the types of factory models used for loading assembly sequences to generate the optimised assembly plans.
- Section 6.7 gives an overview of the genetic algorithm approach for solving combinatorial optimisation problems (NP-hard class), and shows how the method can be adapted to the issue of assembly line balancing.
- Section 6.8 describes the genetic algorithm-method used to balance assembly lines, and thus generate optimal assembly plans.
- Finally, Section 6.9 pulls together all the conclusions drawn from this Chapter.

6.2 Definition of Problem

The assembly line balancing problem entails assembly operation assignment in an ordered sequence, to a set of workstations, such that the precedence relations among the assembly operations are satisfied, and some assembly line performance measure(s) optimised. When considering such problems, the conventional objectives to achieve are:

- Minimise the number of workstations for a given cycle time. This leads to a decrease in the human and technical resources required for operating the assembly line. Thus, it decreases the operational cost of an assembly line.

- Minimise the cycle time for a given number of workstations. A decrease in the cycle time for each workstation results in an over all decrease in the assembly time. This corresponds to a decrease in production cost.
- Minimise the difference amongst workstation times. This involves balancing the workload distribution with respect workstations in use. It improves and/or maintains the workflow on an assembly line by decreasing the idle time on any one workstation. Also, a sense of equality amongst workers on the assembly line is promoted if the workload is evenly distributed throughout the line. This aids the development of a steady and/or increased production rate for a given assembly line.

For the purpose of this research, the analysis of the assembly line used is based on a conveyor based manual assembly line (see Section 2.7). The performance measures considered for a manual assembly line include;

1. Workload smoothness; refers to the comparative workload distribution amongst workstations. The use of the number of assembly operations assigned to each workstation as a measure of the workload smoothness is inaccurate as different operations have different handling and insertion times. A better measure for workload distribution is the "station variation index"; the workload smoothness is estimated by deriving an estimated variance by taking into account the idle time on each workstation. The aim here is to minimise the idle time on each workstation.
2. Work relatedness; refers to the comparative analysis of the types of operations performed on each workstation. Clearly, the handling time for tools and components is reduced if all operations requiring a particular tool are performed at the same workstation, thus further increasing the skill level of the operator.
3. Precedence relations; refers to the technological sequencing requirements or constraints as the order in which the assembly operation can be performed is limited. Floating and fixed AFCs are used to denote the relative possible positions of assembly operations. In addition to technological constraints, zoning constraints are also considered. A positive zoning constraint means that certain assembly operations are to be performed close together, preferably on the same workstation. A negative zoning constraint indicates that assembly operation

might interfere with one another and should therefore not be located in close proximity.

Whilst this issue has already been taken into consideration when generating the optimal assembly sequence (which is loaded on an ideal assembly line to create the ideal assembly plan used as the input for the optimisation process using genetic algorithms), the feasibility of the assembly plans generated has to be maintained through out the optimisation. Hence this check is preformed again. As the pattern of a near optimal assembly sequence is already know, it is easier to recognize when a near optimal assembly plan has been attained.

4. Worker allocation; refers to the effects of adding an operator to an existing workstation. Whilst operation-sharing encourages simultaneous execution of assembly operation and thus overcoming to a certain extent the problem of workload distribution in an assembly, this could also lead to increased idle time on a given workstation.

5. Assembly line flexibility; an assembly system has a high level of flexibility when balance delay, workload relatedness, and workload smoothness are all taken into account when balancing an assembly line. All these three factors increase the capabilities of assembly lines.

6.3 Assumptions

This chapter presents a method for balancing single-model (one product) assembly lines based on the optimisation of the derived functions for the above performance measures. The analysis is performed with the following assumptions made:

1. Unless otherwise stated all assembly operations are performed sequentially. Simultaneously loading of assembly operations are only considered when investigating the possible merits of worker allocation on a given assembly line.

2. An optimised assembly sequence has been generated and is used as the input sequence for the analysis. The optimised assembly sequence is loaded on an ideal assembly line. This acts as the initial assembly plan loaded used as the input to the genetic algorithm.

3. All operators have the same skill level.

4. Unless otherwise stated each workstation is assigned one operator.

6.4 Equations for performance measures of assembly lines

The following terms and functions are used to define line balancing terminology and objective functions respectively for the optimisation of assembly lines.

1. Total assembly time; total time required to assemble the product, it includes transportation time between workstations.

$$T_{wc} = \sum_{i=1}^{m} T_i \qquad \textbf{Equation 6-1}$$

Where,

T_{wc} is the total assembly time on an assembly line

T_i is the assembly task time for task i

m is the number of assembly tasks

i is the number of an assembly task

2. Theoretical cycle time; as derived using the ideal production rate.

$$T_c = \frac{1}{R_c} \qquad \textbf{Equation 6-2}$$

Where,

T_c is the ideal or theoretical cycle time

R_c is the given production rate, supplied by the user

3. Theoretical minimum number of workstations $n_{\min}$, is obtained from a possible set of minimum number of workstations $\{n_{ej}, n_c, n_{hf}\}$, given by Equation 6-3. The theoretical minimum number of workstation is calculated by first evaluating, and selecting the largest possible workstation cycle time from the set $\{T_{ej}, T_c, T_{hf}\}$ (Equation 6-7 and 6-8), and then, using this value to divide the total assembly operation time (Equation 6-4 and 6-5).

$$n_{\min} = \min\{n_{ej}, n_{wc}, n_{hf}\} \qquad \textbf{Equation 6-3}$$

$$n_{ej} = \frac{T_{wc}}{T_{ej}} \qquad \textbf{Equation 6-4}$$

$$n_{wc} = \frac{T_{wc}}{T_c} \qquad \textbf{Equation 6-5}$$

$$n_{hf} = \text{number of assembly tasks for which } T_i \geq \frac{T_{wc}}{2} \qquad \textbf{Equation 6-6}$$

$$T_{ej} = \max(T_i) + T_{j,j+1} \qquad \textbf{Equation 6-7}$$

$$T_{hf} = \frac{T_{wc}}{2} \qquad \textbf{Equation 6-8}$$

Where,

T_{wc} is the total assembly time on an assembly line

T_c is the ideal cycle time, calculated from the production rate

T_{ej} is the minimum cycle time

T_{hf} is half the total assembly time of an assembly line

T_i is the assembly task time for task i

$T_{j,j+1}$ is the transfer time between workstations j and $j+1$

n_{min} is theoretical minimum number of workstations

n_{ej} is the number of workstations, using a T_{ej} as cycle time

n_{hf} is the number of tasks with their task times greater than half T_c

4. Balance delay d, measured as a percentage, is a measure of the line inefficiency, which results from the idle time due to imperfect allocation of work among stations (Groover, 1987).

$$d = \frac{nT_c - T_{wc}}{nT_c} \times 100 \qquad \textbf{Equation 6-9}$$

Where,

n is the number of workstations in an assembly line

T_c is the ideal or theoretical cycle time

T_{wc} is the total assembly time on an assembly line

5. Station variation index σ, measured in seconds, is described as the standard deviation of workstation times; it measures the degree of variation between workstations (Groover, 1987).

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_s - T_c)^2} \qquad \textbf{Equation 6-10}$$

Where,

n is the number of workstations in an assembly line

T_s is the sum of element times at a workstation on an assembly line

T_c is the ideal or theoretical cycle time

6. Index of work relatedness δ, work relatedness requires finding an assignment of operation such that interrelated operations are allocated to the same workstation as much as possible (Groover, 1987).

$$\delta = \frac{n}{\sum_{i=1}^{n}\left[\frac{number\ of\ sequenced\ tasks}{total\ number\ of\ tasks}\right]} \qquad \textbf{Equation 6-11}$$

Where,

n is the number of workstations in an assembly line

number of sequenced tasks refers to the number of identical assembly operations type assigned sequentially to a given workstation. For example, three threaded AFCs loaded sequentially on workstation *i*.

total number of tasks refers to the total number of assembly task assigned to workstation *i*.

7. Total production cost per unit, C_{pc}, is given by Equation 6-12. It is assumed that one of the key deciding factors in the design/redesign of a product is based around the cost per unit. It takes into consideration the material, labour and non-operation cost. Here, non-operation cost takes into account factory (and other overhead cost) and transportation cost only. Other external factors that affect the total assembly cost that are not accounted for include external factors such as holding and transportation cost. The equation presented below (Groover, 1987) has been adapted to suit SALB problems. For the purpose of this research, the material costs has been reduced to zero as the labour cost and other overhead cost are significantly higher than material cost in the case of the product used for the analysis, as discussed in Chapter 7. For the purpose of this research the cost is measured in pounds (£).

$$C_{pc} = C_m + n\left(C_o T_p + C_{no}\right) \qquad \textbf{Equation 6-12}$$

$$C_{no} = Factory\ cost + \left(n_Q C_{BT}\right) \qquad \textbf{Equation 6-13}$$

$$T_p = \left\{T_c, T_{ej}, T_{hf}\right\} \qquad \textbf{Equation 6-14}$$

Where,

C_{BT} is the transportation cost per batch

n_Q is the number of batches

C_m is the material cost per unit production

C_{no} is the non-operation cost (overhead, transportation, internal handling and storage cost)

C_o is the cost rate per operator including overheads

C_{pc} is the total assembly cost per part/product

T_p is the average production time per unit product/part. This value is equivalent to the cycle time for a given assembly line. For the purpose of this analysis, the cycle time can be calculated using Equations 6-2 (T_c), Equation 6-7 (T_{ej}), and Equation 6-8 (T_{hf}). The decision as to which cycle time is used in equation is user dependent.

6.5 Proposed Assembly Line Balancing Method

6.5.1. System Overview

The structure of the assembly line balancing module is shown in Figure 6-1. The aim is to generate optimal assembly plans based on a set of pre-defined performance criteria. The system takes as its input an assembly plan, which is generated by loading the optimised assembly sequence (obtained from the simulated annealing optimisation module) on a manual assembly line. The manual assembly line is created using the green or brown field factory data (see Section 6.6), stored within CAPABLE*Assembly* as flat file databases.

The assembly plan is encoded using a hybrid of the sequence and workstation oriented representation schema (see Section 6.7.1). Through a process of mutation and crossover (see Section 6.7.5), an initial population of assembly plans is created. Each assembly plan in the initial population fitness is evaluated using an objective function (see Section 6.9), which comprises all the performance measures given above (Equations 6-1 – 6-14), to optimise the assembly plans generated based on minimum cycle time, maximum workload relatedness, and maximum workload smoothness, as shown in Figure 6-1.

Each assembly plan is analysed to see if the mutation and crossover process has yielded any assembly plans containing an unfeasible assembly sequence, using some performance test, as shown in Figure 6-1. The best individual in the population, that is the assembly plan with the highest objective function value, is used to create the new population. The process is repeated for a predefined number of generations, or until a convergence is noticed, that is, each individual in the population yields the same objective value.

The assembly plan with the highest objective score, belonging to the final generation population, is the optimal assembly plan. However, all members of the population represent optimised assembly plans, thus providing the assembly planner with a pool of well-optimised assembly plans.

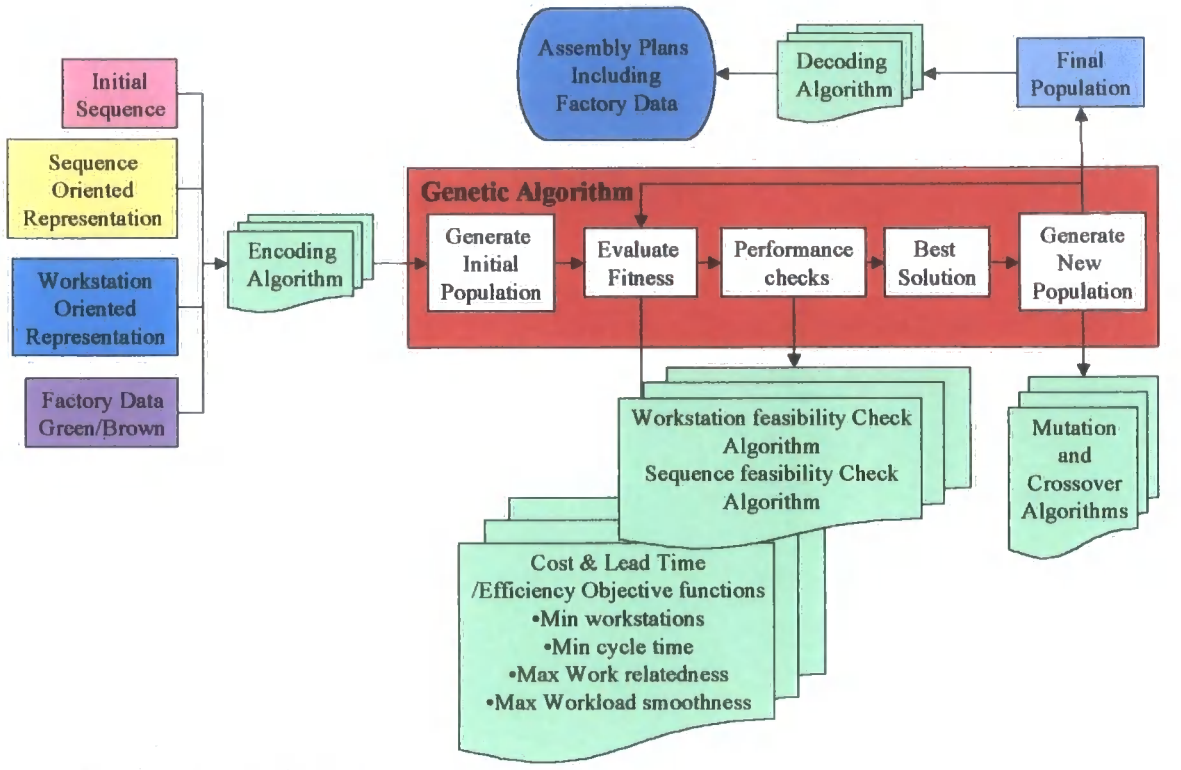


Figure 6-1: Overall module structure

6.6 *Factory model*

The term factory model here is used to describe cells consisting of workstations, tools, and people stored using an object-oriented database. These cells are used to represent manufacturing processes such as machining and assembly.

In the case of assembly, these cells take two formats, cellular assembly units and single model flow lines. For the purpose of this research, mixed model assembly flow lines have not been considered. As with the product model, the factory model is created using an object-oriented method (C++).

A diagram showing the data and hierarchal structure of the factory model generated in CAPABLE*Assembly* is shown in Figure 6-2. As before, the circles represent classes and subclasses. The four main classes used to create the factory model are:

- Factory cells. A factory cell comprises of one or more single model flow lines (assembly lines), and cellular assembly units. The main properties associated to this class include; the number of assembly flow lines in the factory, number of cellular units in the factory, the x and y coordinates of the factory, the x and y

coordinates of the assembly flow lines and cellular units, the number of available assembly flow lines and cellular units available at any given time, and the names of product(s) assembled in the factory.

- Assembly lines. An assembly line comprises one or more workstations. If the assembly line created consists of one assembly workstation, a cellular assembly unit is generated at runtime. If the assembly line consists of two or more assembly workstations, an assembly flow line is generated. The main properties of this class include; the number of workstations assigned to the assembly flow line and cellular unit, the number of available workstations at any given time, the transfer type used to move components and subassemblies between workstations, the transfer rate between the workstations, the x and y coordinates of the assembly line, and the x and y coordinates of the workstations.
- Workstations. A workstation comprises of all the available resources required to carry out an assigned assembly operation. The main properties of this class include; the type of workstation (cellular unit or a flow line workstation), the x and y coordinates of the workstation, and the available resources at the workstation.
- Resources. This class stores all the resources available to workstations within an assembly line. Available resources include tooling requirements, people, machines, stand-alone jigs, fixture information, and storage bins.

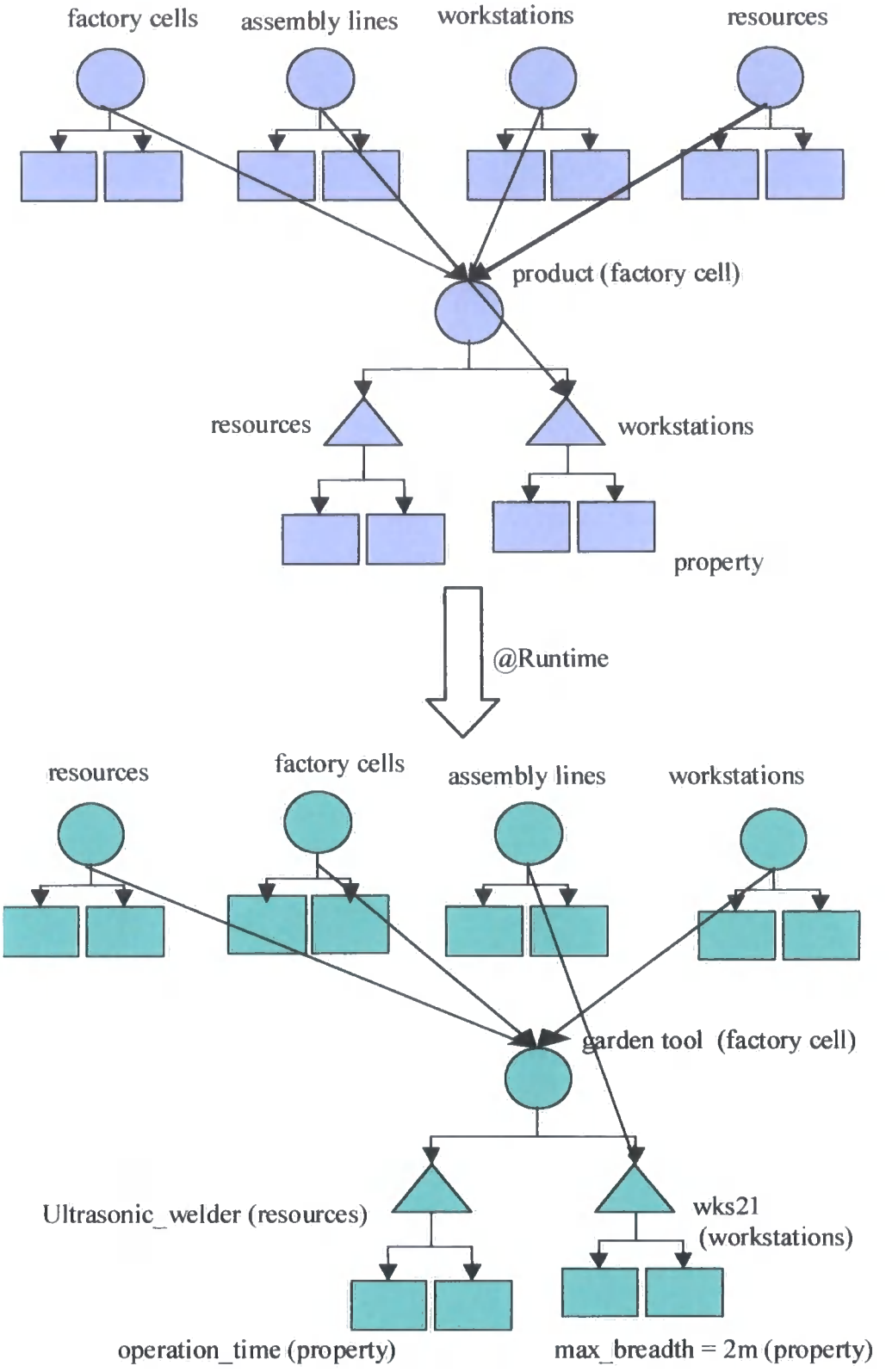


Figure 6-2: Factory model

The triangles represent objects and sub-objects; they are instantiations of classes or specific members. The rectangles are the properties of the objects and the squares can be viewed as values of properties only generated at run-time. When the factory model

is being generated, that is at runtime, the object properties are assigned values stored in the slots shown in Figure 6-2 as squares.

For instance, a product such as a garden tool will have a cell designated for its production with a number of available workstation from a pool of workstations assigned to an assembly flow line. The assembly line will be located within the coordinates set by the product's factory cell and are assumed to be of moving conveyor type. Assembly lines generated within CAPABLE*Assembly* can have continuous, synchronous or asynchronous transfer types.

In assembly lines using a continuous transfer system, the conveyor belt moves continuously at a predetermined speed, stopping for a fixed length of time at each workstation on the assembly line. The operators working on the line have no means of altering the speed or stopping the conveyor belt. Asynchronous assembly lines have predetermined transfer rates between workstations, but the operators on the lines have some means of stopping the conveyor belt. Typically, such systems are fitted with foot pedals or stop/start buttons on the frame of the conveyor belt.

Although continuous transfer is the most common in manual assembly lines, the majority of lines modelled have been assigned asynchronous as they avoid typical problems that are inherently hard to model associated with continuous transfer. This includes incomplete parts when an operator is unable to finish a current part and the next part travels across the conveyor. Each workstation assigned to an assembly line has been allocated a limited amount of resources, unless a green-field (see following section) factory has been generated.

6.6.1. Ideal assembly line and workstation Layout

The assembly lines modelled within CAPABLE*Assembly* are representative of typical assembly lines in industry. The transfer rate from one workstation to another was presumed to be 4.5m/s, in line with the majority of manual assembly lines in industry.

The optimised modelled exhibits a buffer system see Figure 6-3. It should be noted that in an ideal assembly system, a buffer system would not be needed, as a more streamlined design would be preferred. However, in practice, a buffer system has been proven to be more efficient for high volume production.

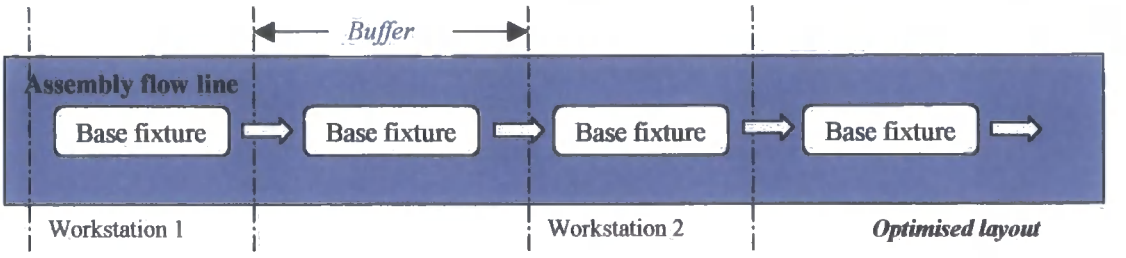


Figure 6-3:Current and optimised buffering systems

The assembly workstations have been designed using ideal space, tooling and ergonomic conditions. The estimated part handling times are greatly dependent on the layout of the assembly workstations. The first step in designing a new workstation is to determine the physiological area of reach for an operator, see Figure 6-4.

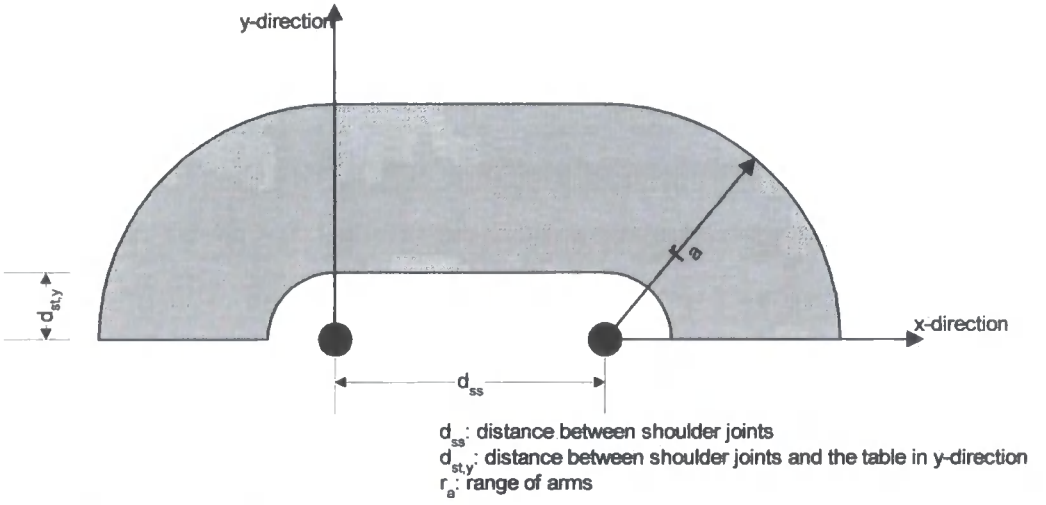


Figure 6-4:Representation of physiological area of reach (view from the top)

CAPABLE*Assembly* uses the dimensions of the fifth percentile of women. They are with reference to Figure 6-4 (see DIN 33402 part 2, 1986; DIN 33406 1988; DIN 33416 1985):

1. d_{ss} = 307mm
2. $d_{st,y}$ = 119mm
3. r_a = 582mm

The values are used as a guide for the choosing and placing the workstation elements (bins and tools). To achieve a reduction in the operation time of standard assembly operations (especially the grasping of parts) performed with each workstation the following strategies were used:

1. Small parts should be placed in the optimal visual area.
2. Parts needed several times should be provided together.

3. All parts are placed within 60% of the maximum area of reach (arms reach, r_a).

Other assumptions made in the optimisation of assembly workstations include

1. Consideration of the visual field when positioning small or difficult to handle parts, that is a time penalty is added on the basic time to perform an assembly operation if the part is deemed to be small or posses handling difficulties such as sharp edges.
2. Same or similar parts are provided together.
3. Simultaneous work of both hands is assumed.
4. All operators are skilled.
5. All operators maintain an upright position.

6.6.2. Green and Brown field assembly lines

All assembly lines generated within CAPABLE*Assembly* can be done in two modes namely brown and green field. Brown field refers to assembly lines generated based on a set of pre-defined available resources. This information is stored in databases and is automatically loaded when a specific product assembly line is generated. Limiting available resources include; number of operations per workstation and workstations that are tool specific. When a brown field assembly line is used assembly operations can only be loaded on workstations capable of performing the operation and have enough space to hold the mating parts.

Green field models refer to assembly lines with no given limitations on resources and space, the system effectively creates a new assembly line. Resources, human and otherwise are loaded as needed and the spatial and geometrical limitations are modified as the operations are loaded on workstations.

6.7 Genetic algorithms

The use of genetic algorithms to solve combinatorial optimisation problems has been developed over the last forty years, notably by Goldberg et al, (1989). Genetic algorithms seek to breed good solutions to the complex problems by a paradigm that mimics evolution. The process is characterised by the initial construction of a population of initial solutions. Solutions in the population mate and bear offspring solutions in the next generation. These reproduction and crossover operations are programmed to replicate the paradigm of survival-of-the-fittest. Over many generations, the solution of the individual members of the population improves; the genetic algorithm terminates when a preset condition (usually a given maximum number of

generations, goodness-of-best-solution, convergence-of-population or any problem specific criterion) has been met and the final population generated should hopefully comprise near optimal solutions. It is important to note that genetic algorithms do not guarantee that optimal solutions are attained.

Over the past four decades the issue of single-model assembly line balancing has been researched extensively using various computational methods (Talbot et al, 1986; Ghosh and Gagnon, 1989; Anderson and Ferris, 1994, Leu et al, 1994 and Suresh et al, 1996). Such methods can broadly be categorised into two groups; those that seek to solve the problem exactly, such as exhaustive search methods (this includes backtracking and branch-and-bound methods), and approximation algorithms (Gerez, 1999). Typically, such methods are applied to situations where the problem size is small or the nature of the solution space is known or can be derived. Both techniques (exhaustive search and approximation algorithms) are particularly problem specific and require a great deal of design and manufacturing information. Such information may or may not be available at the earlier stages of design. Consequently, the robustness of such systems is questionable.

In choosing the most appropriate method to solve intractable problems one of the most important factors is the solution representation. In the case of SALB problems, a simple linear formation (which can be easily represented using genetic algorithms), capable of storing more complex structures is sufficient. Methods such as backtracking and branch-and-bound typical use task trees, to perform a depth and breadth search exhaustively. For the purpose of this research it was deemed imperative to be able to keep the format of the AFCs loaded on the workstations, when encoding the solution. Each AFC is directly liked to the connectivity model and the product model, which simplifies (and speeds ups) the process of decoding and evaluating a feasible solution. Standard schema, inherent to genetic algorithms, can be used to intuitively map the SALB problem. For example, a chromosome can be used to represent an assembly line and the gene the assembly operation assigned to a workstation, in genetic terms an allele.

For the problem at hand, exhaustive searches are not the most efficient method of finding an optimal solution. Branch-and-bound and backtracking methods typically start with a partial solution in which as many variables as possible are left unspecified; values are then systematically assigned to each variable until a fully specified solution (feasible solution) is obtained. Although the cost function or goodness of each solution

is only evaluated for feasible/fully specified solutions, a large degree of fruitless search is still performed; this is expensive in computational terms. Genetic algorithms are good at identifying the aspects of a good solution using tools such as partial match cross over, where patterns/themes are effectively transferred to subsequent generations. Exhaustive methods are simply good at spotting good solutions; this does ensure the best possible solution is eventually attained.

The other group of computational methods is that of general-purpose heuristics, which do not guarantee an optimal solution, but they do however yield near optimal solutions. Genetic algorithms fall into this group of computational methods. The appeal of genetic algorithms lies in the fact it does not require a multitude of mathematical stipulations about the optimisation problem. Due to its evolutionary nature, genetic algorithms will search for solutions without regard to the specific inner workings of the problem. It is capable of handling any objective function (linear or non-linear) and any number of problem specific applied constraints. Thus, genetic algorithms can be used to facilitate the generation of optimal/good assembly plans at the aggregate level of design (Maropoulos, 1995).

Genetic algorithms also have other distinct advantages over the other computational methods within its group, such as simulated annealing and tabu search. These include the effectiveness of genetic algorithms to perform a global search over the solution space due to the ergodicity of the genetic operators (see Section 6.7.5). Methods such as simulated annealing, as discussed in chapter 5, perform convergent stepwise procedures comparing the value of nearby solutions and move to the relative optimum solution. The flexibility of genetic algorithms can be capitalised by using domain-specific heuristics for the effective solution of a specific problem.

6.8 Generation of assembly plans using GAs

The logic of genetic algorithms can be applied to the generation and optimisation of assembly plans. For a given population size (number of individuals in a population) each gene (assembly operation) within the genome (assembly plan) uses the allele[1] set to ascertain its value, each gene is randomly assigned (using a random number generator) an allele from the allele set (set of workstation). Each genome has a length equal to the number of objects stored within the array, that is the number of assembly operations to be assigned to the workstations. The fitness of each genome (assembly

[1] As with genetics, an allele refers to the value assigned to an element/gene within a chromosome. See Section 6.8.2.

plan) is evaluated using an objective function. Depending on the chosen genetic operators (see Section 6.8.1), the best genomes are used to create the next population. The process is repeated until convergence is attained resulting in a final population consisting of only genomes (assembly plans) with high objective scores. The genome (assembly plan) with the highest objective score is deemed to be the best solution (optimal assembly plan).

Two types of genetic algorithms namely, *simple* and *steady state* genetic algorithms were used to solve a simple SALB problem, with minimal restrictions consisting of only ten assembly operations. The difference between the types of genetic algorithms lies in the generation of subsequent populations. Steady state genetic algorithms utilise overlapping populations; a clone of the initial population is generated, for each generation a temporary population is then created. These two populations are then added together, the worst members of the resulting population are removed in order to return the population to its original size. Simple genetic algorithms do not involve overlapping populations; and for each generation the new population is created by mating members of the initial population. The results obtained showed that the simple genetic algorithm, although capable of finding optimal solutions, took longer to converge and was not consistent in finding an optimal solution. As there was no reason to believe this pattern would only be limited to simple product models, the steady state genetic algorithm is used for all simulations. Hence, for the purpose of this research the steady state genetic algorithm is used for all analysis performed.

6.8.1. Genetic Operators

6.8.1.1 Crossover

Crossover is the main genetic operator; it operates on two chromosomes (solutions) at a time and generates an offspring by combining both chromosomes' features. The crossover rate is defined as the ratio of the number of offspring produced in each population to the population size. The ratio controls the expected number of chromosomes to undergo the crossover operation. A high crossover rate allows the exploration of more of the solution space and thus reduces the chances of converging at a false optimum (Gen and Cheng, 1997). However, if the value is too high it leads to a waste of computation time.

To ascertain the behaviour of the crossover operators considered, a set of standard crossover operators traditionally used for combinatorial optimisation was used initially to solve a simple assembly line balancing problem. As before, the operation considered

consisted of ten assembly operations performed on three workstations. The optimisation of the assembly line is based on Equation 6-9 (balance delay, this is also the line efficiency), Equation 6-10 (station variation index), and Equation 6-11 (work relatedness). The cycle time is calculated by dividing the total assembly time by the number of workstations (Equation 6-4).

Crossover operators used includes:

- Partial match crossover (PMX): This operation was suggested by Goldberg and Lingle (1995), and is aimed at maintaining inheritance of adjacency and relative order of elements in the solution structure. An illustration of how the operators considered work is shown in Figure 6-5. A sub-string from Parent 1 is first selected and copied to the same positions in Parent 2, creating Proto-child 2 (shown red in Figure 6-5). Similarly, a sub-string of Parent 2 is copied to the same positions in Parent 1 to create Proto-child 1 (shown in green in Figure 6-5). A pair-wise exchange (Step 3 in Figure 6-5) between the selected sub-strings is performed to obtain a mapping relationship between Parent 1 and Parent 2. This mapping is used to legalise Proto-child 1 and Proto-child 2 to make an exact copy of Parent 1 and Parent 2.

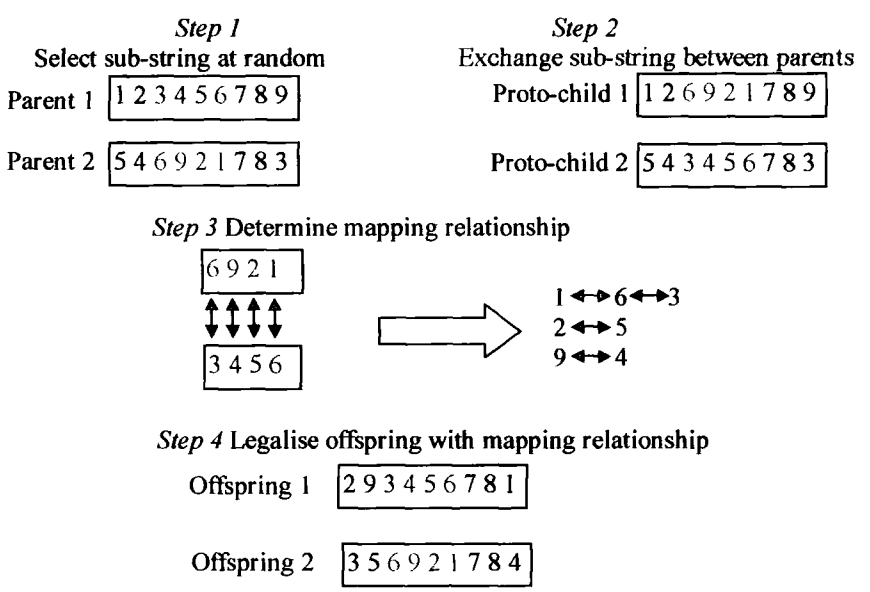


Figure 6-5: Illustration of PMX operator (Gen and Cheng, Genetic algorithms and engineering design, John Wiley & Sons, 1997)

- Uniform crossover. Initially proposed by Syswerda (1989), an offspring is created by randomly selecting an element from each parent. The decision as to which parent the element belongs to is based on the probability of the element coming from each parent. If P_e is the probability of selecting the element from

Parent 1, then $(1-P_e)$ is the probability of selecting the element from Parent 2. A random number is generated x, using a random number generator from the range 0-1. If x is less than P_e (also randomly generated) then the element comes form Parent 1, otherwise the element comes from Parent 2. The element assumes the position in the offspring genome corresponding to its position in its parent genome. This process is repeated until the parents become empty. This operator can be used on genomes of different lengths, but the crossover is truncated to the shorter of the parents and child.

The key benefit of this method is the scope of the solution space it generates, and thus evaluates. However, this approach to generating offspring means it is hard to ensure feasibility of the offspring generated.

- One point crossover. This method of crossover creates an offspring by selecting a crossover point common to both parents and swapping the tail end of each parent as shown in Figure 6-6. In Figure 6-6 a parent genome is made of a sequence of assembly operations assigned to a workstation. Each square represents a gene (assembly operation), which has an allele value (workstation assembly operation has been assigned to). The crossover point is chosen by searching both parent genomes for coinciding gene values (alleles). If such a position cannot be found the mid point of the genome is used as the crossover point. The genes in the tail end of Parent 2 are randomly deleted from the Parent 1 genome. The remaining genes in Parent 1 are used to create the head of Offspring 1 as shown in Figure 6-6. The genes of the tail end of Parent 2 are used to fill up the empty slot lefts in the Offspring 1 genome. The reverse process is repeated to create Offspring 2.

  This type of crossover is useful because it allows for both local search and the exploration of new solutions space. If the crossover point is close to the last gene in the genome, a local search is performed, as the majority of genes within the offspring will retain their original position. If the crossover point is moved towards the first gene, the offspring created will have little resemblance to its parents, thus, providing a new search space. However, this advantage is also its disadvantage, as the methods will be prone finding regions of local optima.

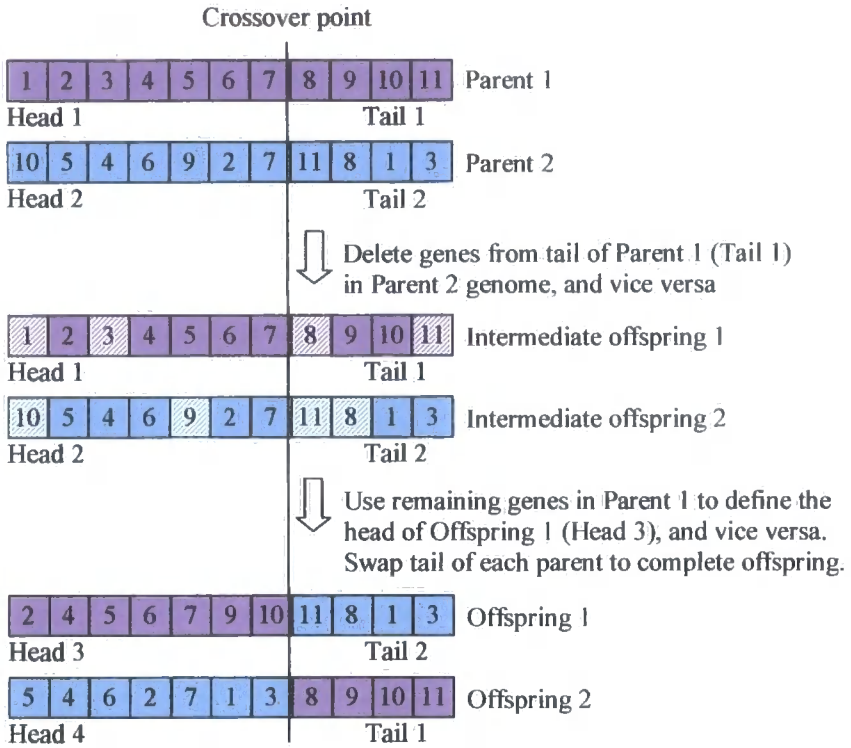


Figure 6-6: Illustration of one point crossover

- Two point crossover. The two point crossover attempts to reduce the probability of getting trapped in the local optima, whilst maintaining the advantage of the one point crossover. The generation of offspring in two point crossover follows the same pattern as the one point crossover, described in Figure 6-6. In this case a second crossover point is also introduced, using the same methods as described above. As before, if coinciding alleles cannot be found, the genome is split evenly to determine the two crossover points. In this case each genome has two tail ends and one central head, as shown in Figure 6-7. The genes in the tail ends of Parent 2 are removed from Parent 1, and the remaining genes are used to create the head of Offspring 1. The tail ends of Parent 2 are used to fill the tail ends of Offspring 1 genome as shown in Figure 6-7. The same process is used to create Offspring 2.

  Whilst this methods does not significantly decrease the probability of finding local optima, it significantly increases the chances of getting out of local optima and thus has a relatively smaller convergence time when compared to the one point crossover. This approach has a greater probability of generating solutions that are not feasible, compare to the one point crossover.

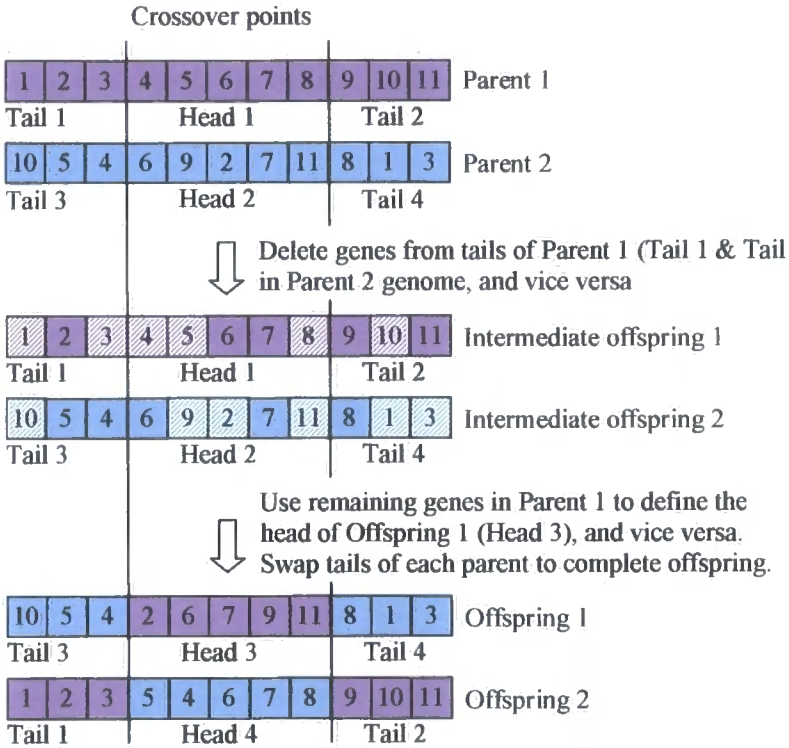


Figure 6-7:Illustration of two point crossover

- Even and Odd crossover. For even crossover, the 0^{th} gene and every other one after that is taken from Parent 1 genome, the 1^{st} and every other gene thereafter is taken from Parent 2 genome to create Offspring 1. Likewise, for odd crossover, the 0^{th} gene and every other one after that is taken from Parent 2, the 1^{st} gene and every other gene thereafter is taken from Parent 1, to create Offspring 2.

  As with the Uniform crossover, although this method does not ensure feasibility in the offspring produced, it does generate and evaluates a large solution space.

The major defect of all the operators considered, excluding the partial match operator, were their tendencies to get trapped in local optimums, hence reducing their reliability. Also, their inability to improve the mean objective score of the individuals within successive populations, and increasing generations resulting lower mean scores for the best population.

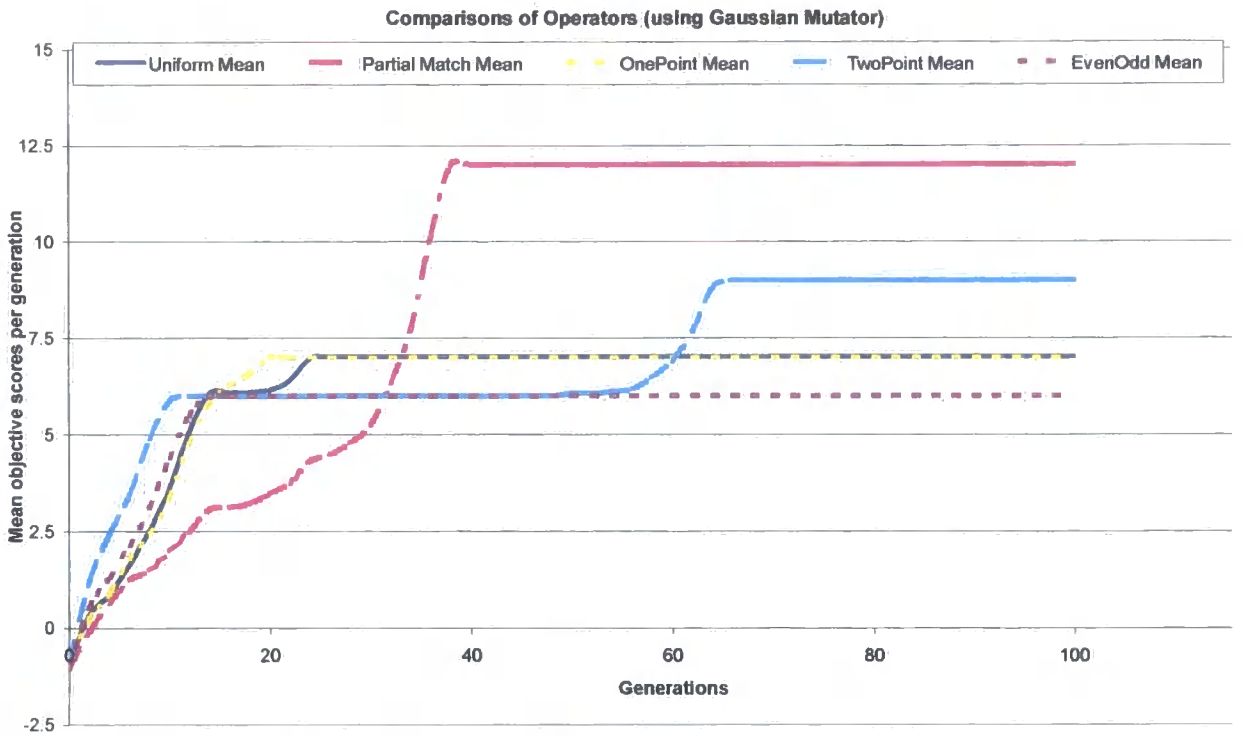


Figure 6-8: Comparison of genetic operators; crossover

The mean values over five separate runs for all the crossover operators used for the line balancing problem is shown in Figure 6-8. The result clearly identifies the partial match operator as superior to all other operators tested, the quality of the successive generations (higher mean objective score values) improves to a greater extent with increasing number of generations and the operator does not appear to exhibit an affinity for immature convergence. The slope of the partial match operator rises steadily and does not continuously generate individuals of similar objective score for more than approximately five generations. All other operators, once in local optima have found it difficult to generate better individuals with higher objective score. The two point crossover does begin to find better individuals after approximately 40 generations. If left to run for sufficient length of time the two point crossover does eventually begin to generate better individuals, although the objective scores were not as high as that attained with the partial match operator. The test was performed over 500 generations, the graph presented in Figure 6-8 have been truncated to reveal the behaviour of the operators when developing earlier generations. The uniform, one point, and even and odd operators did not produce better individuals within this length of time.

The results obtained proved to be consistent with other research (Kim et al, 1996) performed on assembly line balancing. Consequently, the PMX operator was used for the purpose of this research.

6.8.1.2 Mutation

Mutation is a background operator, which produces spontaneous random changes in the population. It operates on one parent, and creates an offspring that is different from the parent. The offspring generated does not normally represent a feasible sequence. Typically, two genes within the genome are randomly selected and swapped to create an offspring, this is known as swap mutation.

The mutation rate is a percentage of the total number of genes in the population. The mutation rate controls the rate at which new genes are introduced to the population for trial. If low, many genes that may have been useful would have never been tried out, but higher values lead to too much random perturbation, in which case the offspring will start losing resemblance to the parents, and the algorithm will cease to learn from the history of the search operator (Gen and Cheng, 1997).

As before, the simple assembly line balancing problem comprising ten assembly operations and three workstations was used for this analysis. In a bid to retain some form of feasibility, two different extensions of the swap mutation were investigated to determine which operator would yield better results, namely:

- Gaussian mutation. Gaussian mutation creates a new offspring genome from a parent genome by assigning a new allele value (workstation number) for a randomly selected gene (assembly operation) within the parent genome. The new allele value is obtained from an allele set of all possible allele values (a set of all possible assembly workstations) using a gaussian distribution based around the current value.

  For example, if the allele set contains 5 workstations {0,1,2,3,4}, and the gene randomly selected [for mutation] has an allele value of 3, its new value is more likely to be 2 or 4, as opposed to 1 or 5.

- Inversion mutations. A description of inversion mutation is shown in Figure 6-9. The operator works by randomly selecting a sub-sting of genes within the parent genome, shown in yellow in Figure 6-9. The allele values of these genes are simply inverted to create the offspring. This form of mutation is possible when using real number allele genomes. In Figure 6-9, the each gene represents an assembly operation, the allele value stored within each gene corresponds to the workstation number the assembly operation has been assigned. It is these values that are swapped. Hence, the third assembly operation (red square in Figure 6-9)

originally assigned to workstation number 0, in the offspring generated is assigned to workstation number 2, as shown in Figure 6-9.

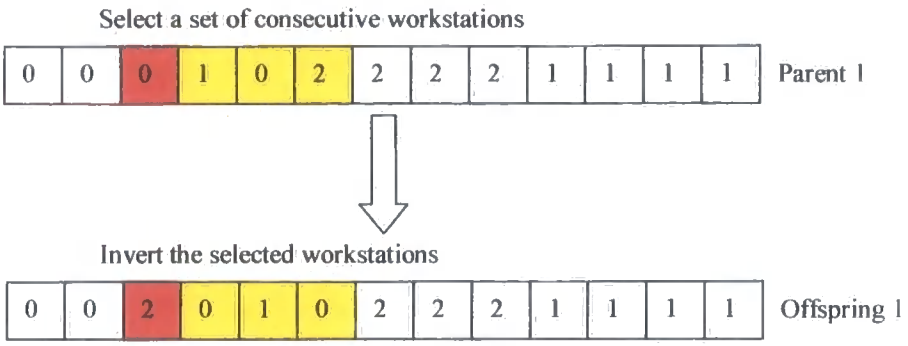


Figure 6-9: Inversion Mutation

A significant difference between the two operators in terms of the objective score was not noticed, as shown in Figure 6-10. However, inversion mutation appeared to produce slightly better results with regards to consistency and objective scores. The preliminary trials showed that a maximum value for the mutation rate was 0.3; higher rates of mutation severely decreased the objective scores of successive populations, increasing the computational time before convergence was attained. In cases of mutation rates above 0.5, convergence was not achieved in 500 generations.

The results obtained are not explicitly consistent with other research in assembly line balancing/assembly planning, as detailed comparison of the mutation operator has not been performed to the same extent as the crossover operator. However, it does attest the well-accepted view that the mutation operator has a somewhat secondary role in genetic algorithms (Goldberg, 1989). Indeed some research (Roach and Nagi, 1996) chooses to ignore this operator.

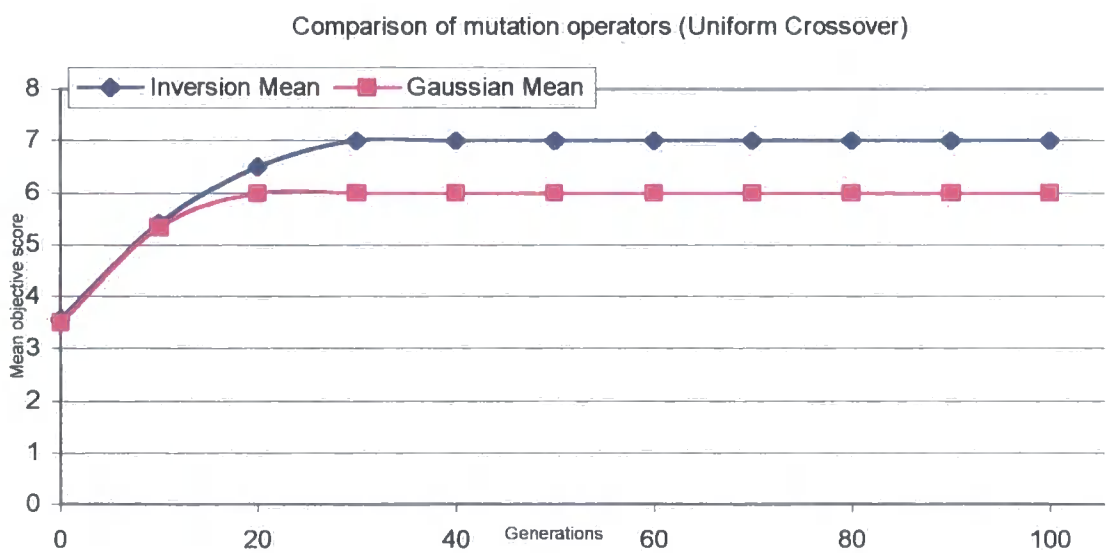


Figure 6-10: Comparison of genetic operators; mutation

Based on the results obtained, an inversion mutation was used with the mutation rate set to 0.3. This was to preserve and increase the diversity of solutions evaluated, without significantly hampering the computational time and feasibility of offspring generated.

6.8.2. Representation

Essentially, genetic algorithms work on the coding and solution space alternately; the genetic operators work on the coding space that is chromosomes/genomes, while evaluation and selection is performed on the solution space as shown in Figure 6-11. Hence a form of representation (encoding and decoding) is required to create a link (mapping) between the two spaces.

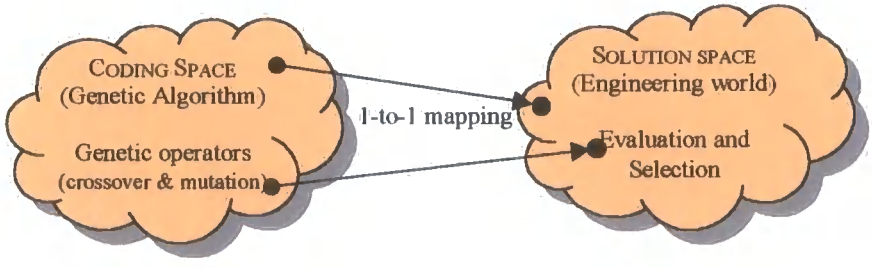


Figure 6-11: Coding and solution space

The first step in constructing a genetic algorithm is the definition of a genetic representation (encoding) of a good solution to the defined problem. Having a good representation that well describes the problem is crucial. Indeed, the representation is required to hold all the information that completely describes a solution. Adopting the basic terminology of genetics: a chromosome (genome) is an encoding of a solution and is a vector in coding space $\Re^n$; a gene is an element of the chromosomes (vector); an allele is a value taken by that element. For example, $x \in \Re^9$ might be a chromosome, x_5 one of its genes, if $x_5 = 6$ then the fourth gene has allele value of 6.

Representation can take many formats, such as strings, arrays, binary, random keys and trees. There are numerous methods of representation that are applicable to the assembly line balancing problem. Those of interest are:

1. Random keys (Bean, 1994). The random keys representation encodes a solution with random numbers. These values are used as sort keys to decode the solution.
2. Workstation based representation (Anderson and Ferris, 1994). In this case, a genome is constructed such that the number of genes required to build the genome is equivalent to the total number is assembly operations. If an assembly operation in the i^{th} position of the genome is assigned to workstation j, the allele

value of the gene (assembly operation) in the i^{th} position is the workstation number j.

3. Sequence based representation (Leu et al, 1994). Here, all assembly operations are sequentially listed in the order that the operations are assigned to the workstations.

4. Precedence-list-based representation (Davis, 1985), originally designed for scheduling problems, can be adapted for assembly line balancing problems. For an n operation, m workstation assembly line balancing problem, a chromosome consisting of m sub-chromosomes is formed for each workstation, each sub-chromosome consisting of a string of length n.

When considering the possible representations of solutions it is important to acknowledge the following aspects:

1. Lamarckian property. The Lamarckian property of a chromosome concerns the issue of whether the chromosome can pass on its merits to a future population through a common genetic operation. Of the three modes of representation considered, random keys have no Lamarckian property that is; an offspring inherits nothing from its parent. Sequence/job-based representations have Lamarckian property that is; an offspring can inherit goodness from its parent. Machine/workstation based representation and precedence-list-based representations have half Lamarckian property that is; part of the segments inherited from parents refers to the same things as the parents while the remaining part refers to different things.

2. Complexity of the decoder. The complexity of the decoder required for the modes of representation described herein, can be classified into the following levels;

    a. Simple Mapping Relation. Operation/sequence based representation and random keys representation, belong to this class.

    b. Simple Heuristic. Precedence-list-based representation belongs to this class.

    c. Complex Heuristic. Machine/workstation based representation belong to this class.

Each workstation stores the assembly sequences it is to perform. Similarly, each AFC stores the workstation number it has been assigned. The encoding format in terms of computer simulation is described in section 6.8.2.

6.8.3. Encoding and decoding

This chapter uses a workstation-based representation, with a real number genome of length equal to the number of assembly operations to be loaded on the workstations and employs the use of allele sets. As already explained in section 6.8.2, alleles are values adopted by genes constituting a given genome. An allele set is the set of all possible values a gene may assume. For example, using a workstation representation, if the total number of workstations available is 5, the allele set $\Re$, for any given genome is $\Re$ = {0,1,2,3,4}. A gene within the genome may be assigned an allele value corresponding a workstation number between 0 and 4. The allele set is stored using an array format.

The decision to use this method of encoding was based on the flexibility it provides in the design of the genetic algorithm in terms of linking the specific objective functions to derived objects (workstations and assembly operations) for evaluation purposes. The other attractive qualities that are inherent in the other methods (namely, sequence and precedence-list-based representation), in particular precedence of assembly operations, can easily be coded as part of the evaluation process whilst still maintaining a high degree of flexibility in the design of the workstations and the allocation of assembly operation to workstations.

It has been argued (Kim et al, 1996) that workstation-based representation can only be used for problems dealing with minimisation of cycle time and workload smoothness where the number of workstations is pre-specified and not for the minimisation of the number of workstations. This, of course, is based on the mode of implementation, and the assumption that the number of workstation needs to be specified by the user, which may not necessarily be the case.

The issue of minimisation of the number of workstations cannot be dealt with as a singular issue (neither can cycle time) with regards to optimisation of assembly lines, an optimal assembly plan needs to be purpose specific. For example, an assembly line can be balanced using three or five workstations both providing acceptable workload smoothness, cycle time and efficiency levels. The question as to whether an assembly plan is optimised using a three or five-workstation assembly line is therefore based on other limitations and restrictions supplied by the user, such as space and cost.

The use of a workstation-based representation can be used to enhance the characteristics of the issue of minimisation of the number of workstations, and is made more favourable with the adaptations of allele set arrays, as this greatly simplifies the issue of decoding. Decoding heuristics are simplified due to the knowledge that the sequence of the operations used is already optimised, thus reducing the search space as well as providing a good idea with regards to a 'good' solution. Relevant workstation data are stored in workstation objects (which are stored as allele sets), so the decoding issue can now be solved using a simple mapping method as shown in Figure 6-12.

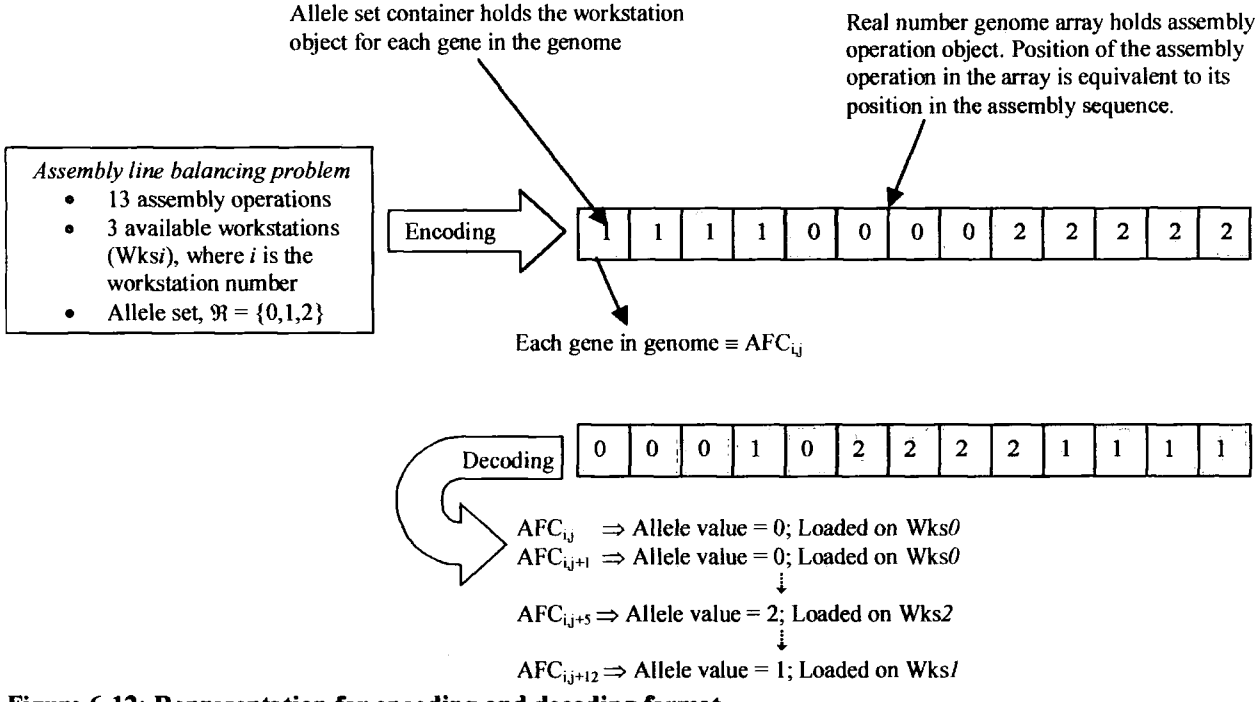


Figure 6-12: Representation for encoding and decoding format

The encoding format is based on the use of allele set arrays using a real number genome. Allele sets ℜ, can be viewed as containers for different values that a gene can assume, it can contain objects of any kind. For the purpose of this research the allele set contains workstation objects as described in Section 6.6. This genome uses an enumerated list of alleles. Each allele is explicitly added to the allele set and any element (gene) of the genome may assume the value of any member of the allele set. In using allele sets as the encoding format, the value of an element corresponds to the workstation the element/assembly operation has been loaded on. Each gene, shown in Figure 6-12 as a green box, in the real number genome represents an AFC object with its own private properties including operation times, precedence ratings and fixed or floating assembly operations, as described in Chapter 4.

For example, if four workstations, ℜ={0,1,2,3}, are required for a given assembly line, each workstation object (allele) is explicitly added to an allowable set of workstations

(allele set), any assembly operation (gene) may be assigned to any workstation (subject to precedence constraints) belonging to the set of workstations generated.

6.8.4. Initial Population and population control

Genetic algorithms are not influenced by the search start point or by the continuity of the search space or by assumptions about convexity. As crossover and mutation operators are controlled by probabilistic parameters, there is no guarantee of finding the same solution in each run. Hence, there is no reason to assume that the performance of a genetic algorithm is enhanced or hindered by using a reselected starting populations or randomly generated starting population. However, some literature (Anderson and Ferris, 1994) advocates the use of random starting population citing improved genetic algorithm performances. For the purpose of this research the initial population of genomes/chromosomes is generated randomly.

As already stated, the approach adopted uses a steady state genetic algorithm for the generation of assembly plans. Here, a certain number of individuals from the initial/current population are passed on to the next population via a selection process. Ideally, the selection process directs the genetic algorithm search towards promising regions of the search space. It is based primarily on the sampling space and the sampling mechanism. The selection method employed for this research is based on a regular sampling space. An in depth discussion of sampling space can be found in 'Genetic Algorithms and Engineering Design' (Gen and Cheng, 1997).

The selection of a member of the current population for migration between generations was initially performed using three sampling methods, namely tournament, roulette wheel, and deterministic sampling, to ascertain the best sampling approach for the analysis. The tournament sampling that gave bad results (low mean objective scores), roulette wheel gave slightly better results (slightly higher mean objective scores), and deterministic sampling generated the best assembly plans (highest mean objective scores obtained). Hence, deterministic sampling is used for selection purposes; it is based on a two-staged selection process. In the first stage, each individual's representation is calculated. The individuals' representation within a population may be obtained from the raw objective scores or from a scaled value based on the objective score. A scaled objective (fitness *f*) score has been used to calculate an individual's representation, expressed in Equation 6-15 (Goldberg, 1989).

$$f = obj\_score - (obj\_ave - c * obj\_dev) \qquad \textbf{Equation 6-15}$$

Where;

obj_score is the raw objective score

obj_ave is the average objective score for the population

c is a small number, used for scaling. The value of c is left to the discretion of the programmer. If the apparent difference between individuals with higher and lower objective scores is to be magnified, this value needs to be greater than 1. For the purpose of this research a value of 1.2 is used.

obj_dev is the objective score standard deviation of the population.

A temporary population is filled using the individuals with the highest expected numbers (fitness scores above the average objective score). Any remaining positions are filled by first sorting the original individuals according to the decimal part of their expected representation, then selecting those highest in the list. The second stage is a uniform random selection from the temporary population. Elitist selection is also utilised, it ensures that the best chromosomes are passed onto the new generation if they are not already selected.

The initial assembly process plan is generated by randomly loading assembly operations from the optimised assembly sequences obtained using the simulated annealing algorithm on workstations of a given assembly line. If the design of the assembly line is not known a green site assembly line is used as described above.

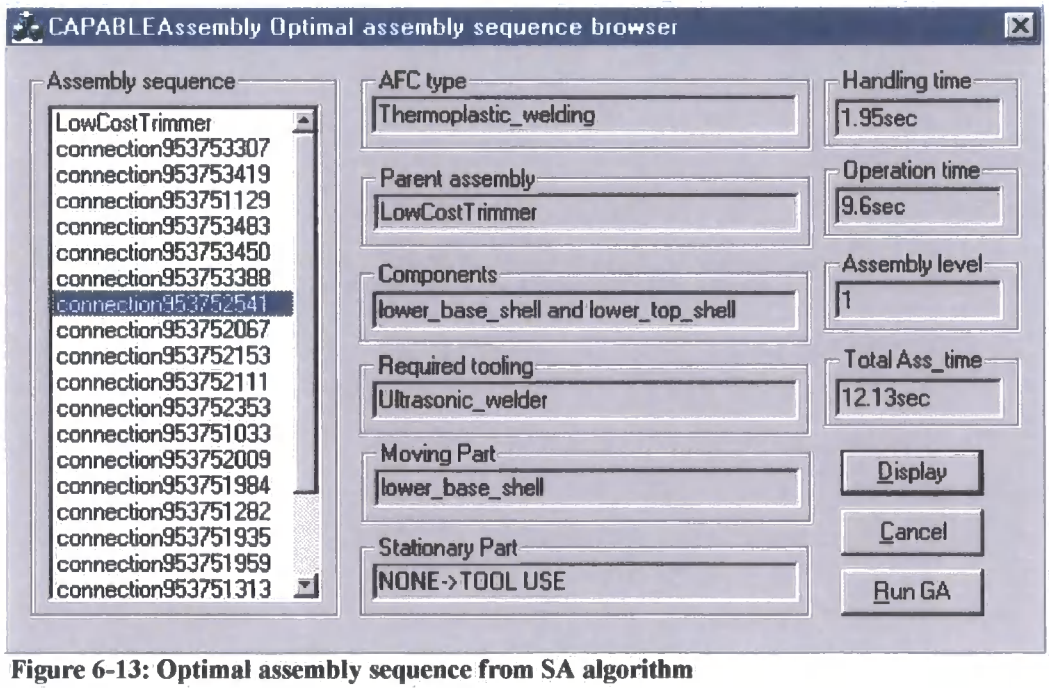


Figure 6-13: Optimal assembly sequence from SA algorithm

6.8.5. Fitness and Evaluation

The fitness of an individual (solution) is equivalent to its objective score derived from the objective function/performance measures. It signifies the ability for a particular individual to survive between generations. The objective score can be used directly as the fitness of an individual. However, it is better to use some form of scaling mechanism to maintain a reasonable differential between relative fitness ratings of chromosomes and to prevent a too-rapid takeover by some super chromosomes. Different scaling methods (Gen and Cheng, 1997) can be applied depending on the problem and the type of values expected. For the purpose of this research, a "sigma truncation" scaling probability has been applied. If the fitness of the individual being evaluated is less than zero, the fitness of the truncated, that is, set to zero. This method was used to deal with negative objective values and to incorporate the problem-dependent information into the mapping process. The mapping from objective score to fitness score for each individual is given by Equation 6-15.

Although all the performance measures listed are considered in the generation of optimal assembly plans, the genetic algorithm is guided essentially by two objective functions, minimisation of number of workstations (Section 6.8.5.1) and minimisation of the cycle time (Section 6.8.5.2). Other factors including balance delay, workload smoothness, work-relatedness and worker allocation are all optimised with every genetic algorithm plan generated.

6.8.5.1 Minimisation of number workstations

The minimisation of the number of workstation is controlled by Equation 6-2. A set of possible values for the minimum number of workstation is calculated using Equations 6-4, 6-5, and 6-6, n_{ej}, n_{wc}, and n_{hf} respectively. The genetic algorithm is looped to generate assembly plans for each value of the theoretical minimum number of workstations from the highest to the lowest within the set. In each case, the result is the best individual (individual with highest objective score) in the final population. This best individual (optimal assembly plan) generated in each genetic algorithm run for each minimum number of workstation over a given number of generations (a value of 500 generations is used for this analysis) is stored, to create a set of optimal assembly plans. This set contains an optimal assembly plan for each number of workstations optimised. For each assembly plan within this set, a total assembly cost per unit is calculated using Equation 6-12. Although all data is provided to the user, the suggested optimal

assembly plan is based on the optimal assembly plan with the least estimated total assembly cost.

Figure 6-14 shows the route and decisions made during the optimisation process.

The GA takes as its input:

- Basic line balancing data; production rate, number of shifts per day, number of hours per shift.
- Total product assembly time. The total product assembly time can be supplied by the user, or is calculated within the algorithm as the summation of assembly operation (AFC) process time. A relaxation factor is also added to the overall assembly time to cater for external factors (such as rejects and human error) encountered when dealing with manual assembly lines. This value is supplied by our industrial collaborator, and is based on the type of operation being performed. Within the algorithm this value is calculated as a percentage (maximum of 4%) of a given assembly operation (AFC).
- Transportation time. This is the time taken to move a subassembly between workstations. A default speed of 4.5m/s, consistent with industry is provided.

With the above input data the algorithm calculates a set of possible minimum number of workstations, as shown in Figure 6-14.

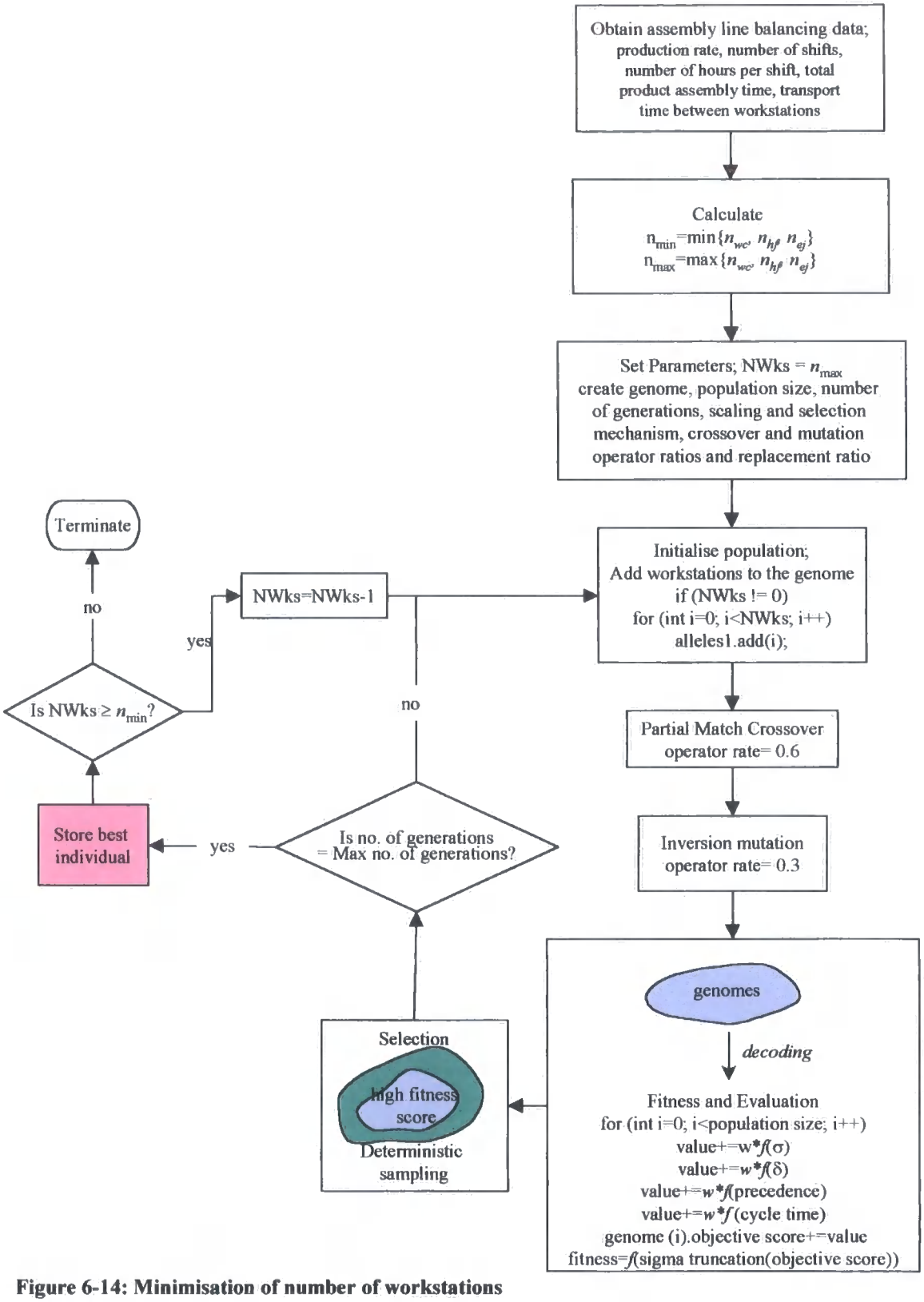


Figure 6-14: Minimisation of number of workstations

The genome is created by loading AFC objects sequentially from the optimised assembly sequence obtained using simulated annealing into each gene of the genome. The number of AFCs within the assembly sequence determines the genome length.

Next, the allele set is generated from the greatest value of the three possible values of minimum number of workstations. For example, if $n_{ej,} = 6$, $n_{wc} = 4$, and $n_{hf} = 2$, the allele set is generated by loading 6 workstation objects into an allele set array for the first GA run. The workstations are obtained from the factory database and the type of workstation loaded is based on whether the assembly line is deemed green or brown field (see Section 6.6.2). This means any AFC can be loaded on a workstation stored in the allele set array.

Once the genome and allele set is created the population can be initialised and the natural selection process begins. The partial match and inversion operators as used as the preferred genetic operators for creating subsequent populations.

Each genome is decoded to calculate its objective score and thus, its fitness. The objective score is calculated using an accumulative process. Each genome starts with a '*value*' of zero, the station variation index σ, for each workstation is calculated using Equation 6-10 and added to this *value*. The index of work relatedness for each workstation is calculated using Equation 6-11 and added to the current *value*.

The feasibility checks (see Section 6.8.6) are also taken into account when evaluating the objective score for each genome. Two functions, *f*(precedence) and *f*(cycle time) are used to ensure assembly operations comply with precedence rules (as described in Section 4.3.2), and that operations loaded on specific workstations can be performed on the assigned workstation respectively. The result of both functions is added to the current *value* of the genome. The total *value* of the genome is stored as the objective score of each genome, as shown in Figure 6-14.

The fitness of each genome in a population is calculated using Equation 6-15; the genomes with high fitness scores are used to create successive populations. A check is made to see if we are in the final generation, if not the next generation is created from the current final population, and the process is repeated (as shown in Figure 6-14). If we are in the last generation, the best genome in the final population is stored, and a check is made to see if the analysis was based on the lowest number of workstations from the possible three values of minimum number of workstations. If not, the current number of workstations is decreased by 1, and the GA is run again, as shown in Figure 6-14, otherwise the GA is terminated. This process results in a set of stored genomes each representing the optimal assembly plans for a given number of workstations.

The total assembly cost for the stored genomes (optimal assembly plans) is calculated using Equation 6-12. The optimal assembly plan is deemed to be the assembly plan with the lowest associated total assembly cost.

6.8.5.2 Minimisation of cycle time

The minimisation of cycle time is controlled by Equation 6-2, by calculating the minimum cycle time T_c, using a given production rate. However, if this information is not available, the minimum cycle time is then calculated as the maximum assembly operation time associated with a single AFC plus the transport time between workstations, T_{ej}. The genetic algorithm follows a similar route to that shown in Figure 6-15. The result in this analysis is the best individual (equivalent to the assembly plan with the highest fitness score) along with the best population. The fitness score of each individual in of a population are stored in order of magnitude from the highest to lowest of the fitness score. For example, the best ten assembly plans would equate to the first ten individuals in the final population.

The flow of the algorithm (shown in Figure 6-15) is described in the preceding section. This algorithm only calculates one value for number of workstations based on the calculated cycle time, T_{wc} or T_{ej}. The objective score for each genome in the population is calculated as described in the preceding section. The genetic algorithm terminates after a pre-specified number of generations, and the total assembly cost is evaluated for the genome (assembly plan) with the highest objective score.

6.8.5.2.1 Worker Allocation

The maximisation of worker allocation is activated when minimisation of cycle time is selected. The system searches for the existence of non-related assembly operations (obtained from the connectivity network) in each workstation. In such cases an additional operator is added to the workstation in question and a new cycle time is calculated. The process is repeated for all workstations, the maximum estimated (with or without an additional operator) cycle time is used as the new cycle time of the SALB problem and the assembly line is balanced using the new cycle time.

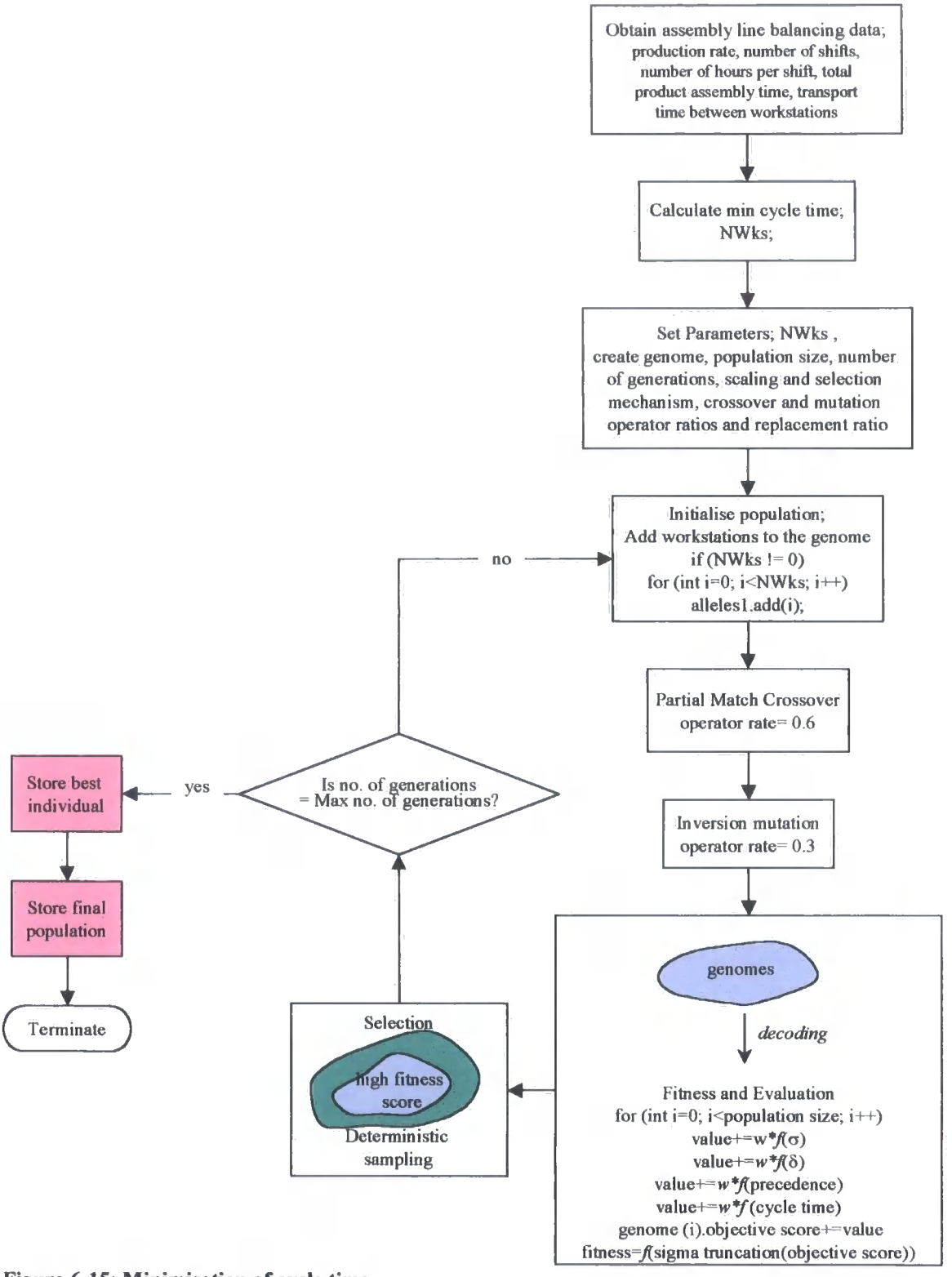


Figure 6-15: Minimisation of cycle time

6.8.6. Feasibility Checks

Each genome undergoes a series of feasibility checks to ensure the feasibility of the assembly plan obtained. These include a workstation feasibility check and a sequence feasibility check.

1. Workstation feasibility check, *f(cycle_time)*. The workstation feasibility check ensures the assembly operations loaded on a particular workstation are within the maximum allowable time per workstation. It also ensures the assembly operation loaded on each workstation can be performed on its designated workstation by checking tool type availability.

   Equation 6-16 gives the function for '*cycle_time*'. The total assembly time for all the assembly operations loaded on a given workstation is evaluated. The workstation is assigned a cycle time index (*CT*) of +1 if this value is less than the workstation cycle time, or -1 if it is greater. The '*cycle_time*' variable of the genome is then the sum of each of these time indices divided by the total number of workstations,as shown in Equation 6-16. This variable has a maximum value of 1.

$$f(cycle\_time) = \frac{\sum_{i=1}^{n} CT_i}{n} \qquad \textbf{Equation 6-16}$$

   Where, *n* is the total number of workstations.

2. Sequence feasibility check, *f(precedence)*. This ensures the assembly sequence loaded on particular workstations is feasible. That is, it checks the relative position of fixed and floating assembly operations and the precedence ratings of the assembly operation assigned to a particular workstation. It also ensures the work-relatedness (Equation 6-11) of assembly operations loaded on each workstation.

   Equation 6-17 gives the function for '*precedence*'. The precedence rating already stored in the AFC object (as a result of the sequence optimisation using SA) is compared for all AFC loaded on a workstation. If two consecutive AFCs follow the rules of precedence (see Section 4.3.2), a value of 1 is added to the '*precedence*' variable of each genome. Otherwise a value of –1 is added to the '*precedence*' variable of the genome. If all assembly operations are in feasible positions within the genome, the maximum possible value for *precedence* is equal to the total number of AFCs in the assembly sequence. Once a value for *precedence* is calculated, it is divided by the total number of AFCs, this gives Equation 6-17 a maximum value of 1.

$$f(precedence) = \frac{\sum_{i=1}^{m}(precedence)}{m}$$ **Equation 6-17**

Where, m is the total number of AFCs.

6.9 Calibration of objective function

The objective score is the summation of the station variation index (σ), index of work relatedness (δ), balance delay (d), and feasibility checks (*cycle_time* and *precedence*). Ideally, one would like to normalise each variable (as done with the simulated annealing algorithms objective function) and equate the sum to the objective score. Unfortunately, due to the nature of the equations used, the variations cannot be mapped quite so easily as it is not possible to estimate the maximum values for σ, and d. The issue is further complicated by the variations between each variable considered.

Take for example the station variation index (Equation 6-10) provided below;

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_s - T_c)^2}$$

For good solutions, the difference between the total assembly operation time on a workstation, and the workstation cycle time (T_s-T_c), will be very small, possibly < 0. As this value is squared, and divided by the number of workstations, even though the square root of this value is eventually obtained, the numerical magnitude of sigma is still very small. When inverted, the result is a relatively [to the other variables considered] large number of good solutions [as desired]. However, this value tends to be too large, and there is a tendency to dwarf the other variables considered, especially the variables for feasibility; *cycle_time* and *precedence*.

The solution to the problem is to formulate expressions for each variable in relation to the station variation index. The station variation was chosen for the two reasons:

1. The station variation index is the most volatile of all the variables.
2. A good (low) value for station variation index, infers a low value for the balance decay will be obtained (Equation 6-9). This in turn indicates the efficiency of the line will be high (balance delay is a measure of the inefficiency of an assembly line).

In its simplest form, the objective score is expressed as shown in Equation 6-18. The function for balance delay and station variation index is inverted to assign higher values to better solutions.

$$obj\_score = \frac{1}{f(d)} + \frac{1}{\sigma} + f(\delta) + f(cycle\_time) + f(precedence) \qquad \textbf{Equation 6-18}$$

On inspection of Equation 6-11, the maximum value for the index of work relatedness is equal to 1. This occurs if all the assembly operations assigned to each workstation are of the same or similar type.

Since the maximum value for *f(cycle_time)*, *f(precedence)*, and $f(\delta)$ is 1, we can use an alternative definition for the objective function, expressed with respect to the station variation index by Equation 6-19. The values generated for balance decay are of sufficient magnitude not to require representation in terms of the station variation index, if it is not expressed as a percentage.

$$obj\_score = \frac{100}{d} + \frac{1}{\sigma} + \frac{\delta}{\sigma} + \frac{f(cycle\_time)}{\sigma} + \frac{f(precedence)}{\sigma} \qquad \textbf{Equation 6-19}$$

Equation 6-19 can be further condensed, but has been left in the format above to mimic the computational process. Also, this format allows users to impose weights on certain/all variables if one feels insufficient emphasis is being place on a specific variable.

6.10 Illustrative example

A number of products were used for testing the feasibility of the method and accuracy of the system. The examples presented herein are in relation to an outdoor, lightweight product. Only the result of the analysis is presented. A detailed step-by-step guide can be found in the following Chapter.

The contact, precedence and technological constraints algorithm were used to generate the corresponding connectivity model from the aggregate product model. The connectivity model was used to generate an initial assembly sequence. This acts as the initial assembly sequence for the simulated annealing algorithm.

An optimal assembly sequence is obtained using the simulated annealing algorithm as described in the preceding Chapter. The analysis was performed using a steady state genetic algorithm with deterministic sampling for population selection, using the following genetic and line balancing parameters:

1. Population size: 110
2. Number of generations: 1000
3. Convergence criteria: Number of generations = 500
4. Genome length: 27

5. Crossover rate: 0.6
6. Mutation rate: 0.3

Assembly line balancing data are:

1. Production rate: 1500
2. Number of shifts per day: 2
3. Number of hours per shift: 8

The two modes of optimisation, minimisation of number of workstations and minimisation of cycle time were performed

1. Minimisation of number of workstations. This section uses a product assembled using 18 AFCs to illustrate this optimisation process.
    a. Number of workstations estimated using production rate (T_c) = 3, cycle time = 38 sec.
    b. Number of workstations estimated using highest single assembly operation time + transfer rate = 6, cycle time = (10.5+4.5) = 14.9 sec.
    c. Number of workstations estimates using number of single assembly operations with an operation time of ≥ half total assembly time = N/A

    The route followed by the genetic algorithm is as discussed in Section 6.8.5.1. The result of genetic algorithm is shown in on Table 6-1. The assembly plan generated is a sequence of numbers, each number represent an allele value (workstation) for a gene (an assembly operation) from the genome (assembly plan). Hence, in the six-workstation set-up, the first six AFCs are loaded on workstation 2, the next three AFCs are loaded on workstation 0, and so on. With the six-workstations set-up, the cycle time per workstation is significantly less than that of the three-workstation set-up, yielding a lower value for the percentage inefficiency (d) of the line. However, although the standard deviation of the line of the six-workstation set-up is higher than that of the three-workstation set-up, it is still within an acceptable realm. However, in today's economy the overriding factor is cost, and unfortunately the six-workstation set-up will prove to be too expensive once shop floor space, labour cost and other overhead cost (including tools) are taken into consideration. The assembly cost per unit is evaluated using Equation 6-12. The three-workstation set-up,

although it provides the lowest distribution of assembly operations in terms of operation times (σ=1.32239), results in a relatively high degree of inefficiency.

Assembly plans generated	Sigma (σ) sec	No. Wks	% Balance delay (d)	Assembly Cost per unit C_{pc} (£)
22222200055521141013133335 34	5.8	6 wks	3.4	9.81
11114414444333333033222222 0	15.9	5 wks	62.3	8.32
11111313330000000023222210 2	10.3	4 wks	29.9	6.66
22222222222211111111120001 00	1.3	3 wks	10.1	4.99

Table 6-1: Results for minimisation of number of workstations

2. Minimisation of cycle time: This section uses a product assembled using 27 AFCs to illustrate this optimisation process.

The result of the genetic algorithm is shown in Figure 6-16.

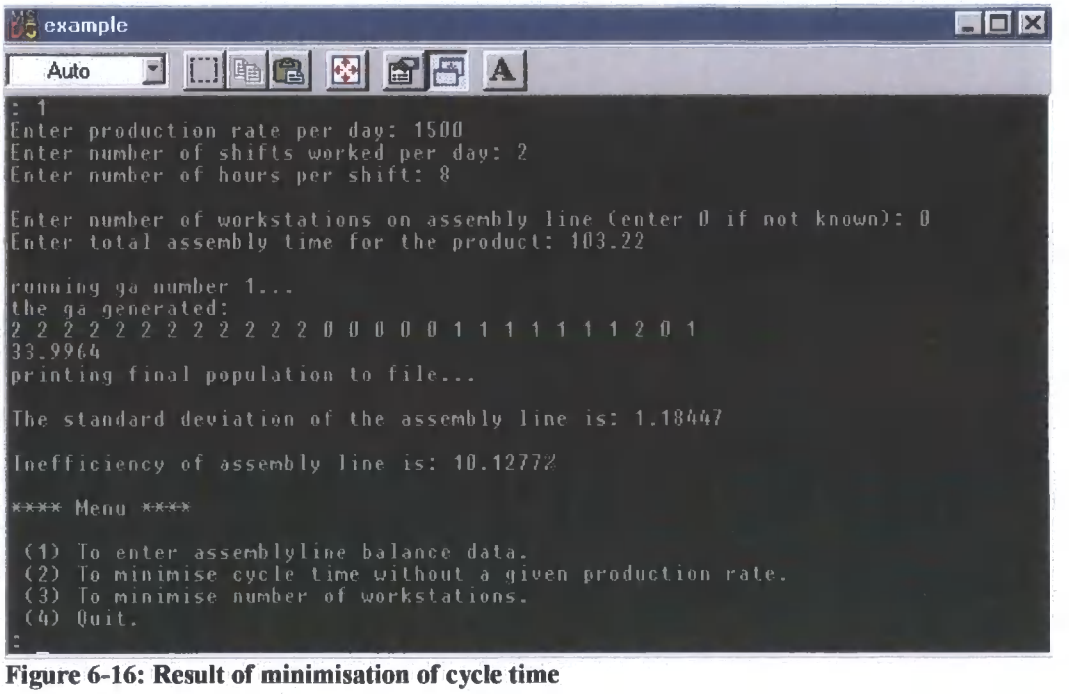


Figure 6-16: Result of minimisation of cycle time

The assembly line balancing data is shown in Figure 6-16. As the system is capable of calculating the number of workstations required, a value of zero can be entered, if the number of workstations is not known. With the line balancing data presented, the cycle time is 38.4 seconds, and the number of workstations = $\frac{103.22}{38.4} = 2.69 \equiv 3$ workstations.

The route followed by the genetic algorithm is as discussed in Section 6.8.5.2. As before, the assembly plan generated is a sequence of numbers, each number represent an allele value (workstation) for a gene (an assembly operation) from the genome (assembly plan). Hence, for the results displayed in Figure 6-16, the

first twelve AFCs are loaded on workstation 2, the next five are loaded on workstation 0, and so on.

From the results presented in Figure 6-16, the suggested assembly plan has a low standard deviation, depicting a satisfactory level of workload distribution between the workstations. The low level of inefficiency shows a solution with minimal idle time across all workstation has been achieved for the assembly plan generated. The precedence relations between assembly operations are also maintained, the majority of operations adhere to the original optimised assembly sequence. The overall objective score of the final population of solutions is shown in Figure 6-17. It can be seen from Figure 6-17, that the solution has converged, with the majority of individuals in the population attaining a high objective score. However, only a few members represent optimal assembly plans.

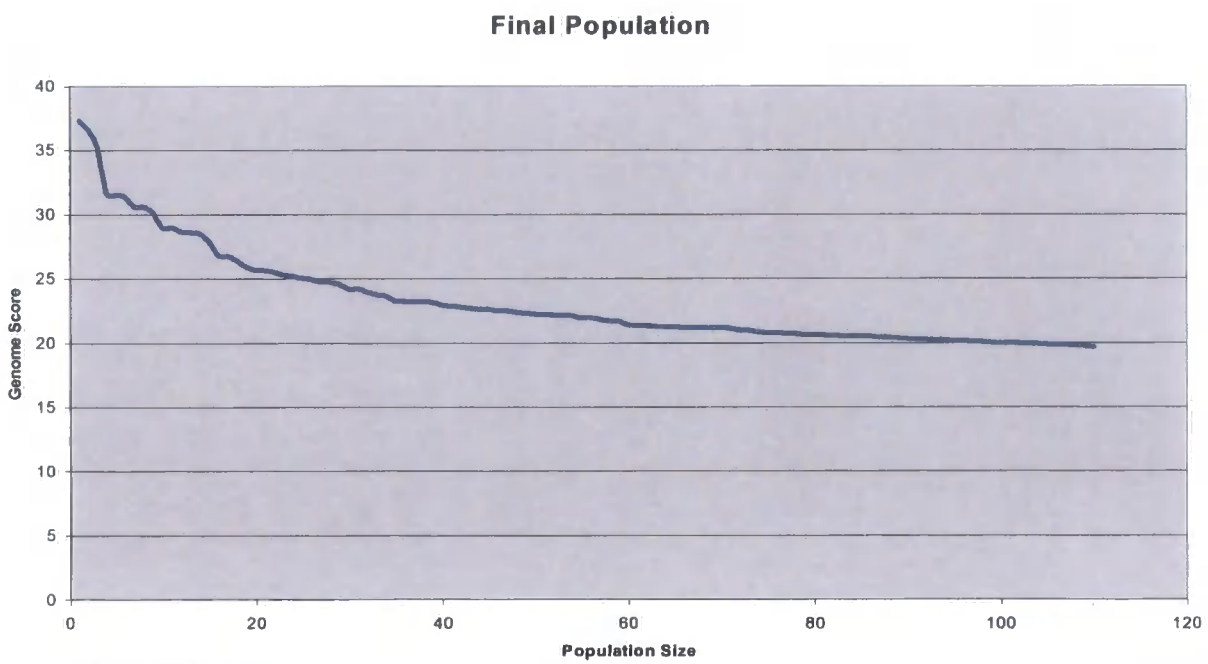


Figure 6-17: Final population

The variables namely the station variation index, the index of work relatedness, precedence, and cycle time, had to be rebalanced during the course of this analysis. It was found that for assemblies with larger number of operations, the station variation index appeared to swamp both the *precedence* the *cycle_time* variable. This lead to the derivation of Equation 6-19, initially, the *precedence* and *cycle_time* variables were balanced with respect to the variance on each of the assembly workstations (the station variation index, is essentially the standard deviation on each workstation).

6.11 Conclusions

A system for the generation of optimal assembly sequences has been presented. The distinction between this system and other methods lies in the way the problem has been designed. It does not seek to simply produce assembly plans based on the minimisation of cycle time or number of workstations, rather it merely uses these parameters for the generation of the initial population, the main interest lies in the 'goodness' of the solutions/assembly plans generated.

The two modes of operation are independent; they use the same genetic algorithm, but are based on two distinct objective functions. The module has been designed through a series of trial and error experimental runs using a large number of products. The system has been balanced in such a way to give significantly higher objective scores to solutions with low station variation index and high values of kept precedence relations. This is achieved by mapping derived weighted power functions to the calculated sigma and delta values giving each performance measure equivalent ratings for similar values. Other measures used including precedence rating and workstation times are subsequently matched to the existing pattern by estimating their respective maximum values and deriving appropriate mapping functions. The result is a well-balanced means of evaluating the 'goodness' of any assembly plan for both initialisation methods (cycle time and number of workstations) of the genetic algorithm.

Ideally, an optimal assembly plan is the result of loading the optimised assembly sequence obtained from the simulated annealing algorithm sequentially on a series of workstations. Hence, an ideal optimal assembly plan is known prior to the optimisation process using genetic algorithms. Whilst the assembly sequence is not altered, the workstations the assembly operations are assigned to changes during the course of the optimisation process. This indirectly changes the assembly sequence, hence feasibility checks, which to some degree overlap with the work already done in generating an optimal assembly sequence, were introduced.

This does not lessen the importance of the simulated annealing algorithm. With the knowledge of an ideal optimised assembly plan, it is easier to see if the genetic algorithm has converged and if, indeed the result is an optimised assembly plan.

7 Testing and Validations

7.1 Introduction

This chapter presents the various scenarios used to test the applicability and validity of CAPABLE*Assembly* based on a series of predefined environmental, logistical, and computational constraints. The aim here is to prove that the basic hypothesis, and logic, behind the methods presented are feasible, and potentially have significant industrial applications.

The evaluation and validation of the concepts proposed in this work is carried out in three stages. Firstly, we conduct a series of experiments to confirm the accuracy of the assembly times that are used to estimate assembly operation times. Next, a series of aggregate assembly process planning tasks are performed using a simplified model of an industrial product. Finally, a significant industrial case study is carried out to demonstrate the effectiveness of the methods developed in theories of product Design for Assembly.

This chapter has the following layout:

- Section 7.2 presents the testing criteria used to verify the validity of the solutions obtained using the methods presented in the preceding Chapters. This includes assembly time generation, assembly sequence generation, and assembly line balancing problems.
- Section 7.3 presents the various experiments used to validate the assembly times used within CAPABLE*Assembly*.
- Section 7.4 sets out a series of industrial case studies using the modelling methods described in Chapter 4, and the optimisation methods put forth in Chapters 5 and 6 to derive optimised assembly plans for four conceptual designs of a given product. The industrial case studies are used to verify the various hypotheses and associated modelling in accordance to the testing objectives documented in Section 7.2.
- Section 7.5 pulls together the possible conclusions that can be drawn from the results of the experiments/scenarios, and the industrial case studies. It presents a critical analysis of the advantages and limitations of the methodology when applied to industrial product Design of Assembly.

7.2 Testing objectives

7.2.1 Validity of solutions

The solutions are tested against the following criteria

1. *Engineering terms*. In this category of criteria, we assess whether the modelling technique (Aggregate product modelling), and the optimisation algorithms (Genetic algorithms and Simulated annealing) presented are:
    a. easily applied to a variety of product model configurations.
    b. suitable for selecting and evaluating assembly processes.
    c. capable of generating assembly sequences and routings that are technically feasible, and realistic.
    d. capable of generating alternative assembly process plans.
    e. capable of producing reasonably accurate estimated assembly times and thus, assembly cost.
    f. easily employed at the conceptual stages of design.
2. *Computing terms*. It is important to note that the requirements in terms of aesthetics, and user interface are not crucial. CAPABLE*Assembly* was essentially produced as a means for testing the algorithms and methods proposed. The development of an integrated system, was beyond the scope of this research. However, the system is required to be suitably robust and reasonably user friendly, providing a detailed assessment of the results of the evaluation. Aggregate assembly process plans are produced within an acceptable time span, and the computational methods used must also converge in an acceptable length of time. Otherwise, the system must fail gracefully. The system is portable, easy to use and understand. The computational methods used are applied within acceptable boundaries. In computing terms the system requires:
    a. communication between software modules, with quick and easy transfer of information to any module.
    b. operation of modules with only partial result. This means that the activities of a module can be started before the completion of the activity of another module (parallel processing). Therefore the interaction

between modules can be really effective, taking advantage of the provision of early feedback.

c. data coherence between different modules. All modules have common user interfaces, data structures and information semantics to avoid errors and inefficiencies.

d. operation of modules by non-experts. All designers must be able to use (albeit in a non-optimised way) the modules where collaborations could influence their specifications, for example a mechanical design engineer must be able to use the assembly planning module to obtain time and cost estimates.

7.2.2 Constraints applied

CAPABLE*Assembly* broadly comprises three main computing functional modules, aggregate product modelling, assembly planning and resource planning. Each module is subject to a series of constraints.

1. Aggregate product modelling; constraints here are imposed to aid the generation of a product model suitable for assembly planning and optimisation. It makes basic assumptions on the shape and structure and composition of the components modelled. Constraints include;

    a. The products are derived principally from prisms, moulds, cylinders and wires.

    b. Assembly features modelled are limited to the assembly features considered for the purpose of this research. These assembly features have been detailed in Appendix D.

    c. Standard assembly operations modelled are limited to 'threaded' (bolts and screws), 'riveting' assembly operations, labelling, pressure fits, packaging, placement, and welding assembly operations.

    d. Derivation of weight of components is performed within the system. The weights are estimated from the positive features used and where applicable negative features have also been taken into consideration. The user can also input the actual weight of the product in the product database.

e. The assembly times associated with the aggregate product model are ideal assembly times for handling and insertion assembly operations. Relaxation values can subsequently be used to modify the assembly times, as has been done during the course of this research. Relaxation values are obtained from the industrial collaborators.

2. Assembly planning. This generates an optimum assembly plan for geometrical, technical and topological information obtained from the aggregate product model. Constraints include:

   a. CAPABLE*Assembly* only takes into consideration reorientation, parallelism and stability of components and subassemblies for assembly sequence generation.

   b. The influence of tolerance on the generation of assembly plans is currently not considered. The product model does not currently have any tangible 3D representation for tolerances.

   c. The estimates for assembly cost are entirely based on the data supplied by the company. This includes data such as production rates and labour cost.

3. Resource planning. This helps the user to produce a logical factory or cell layout by regrouping the operation on cells or workstations, attributing sets of equipment to each of them, while being constrained by a given target cycle time.

   a. CAPABLE*Assembly* solely caters for assembly lines using conveyor belts moving at constant speed between a pre-specified number of manual assembly workstations. The system can be adapted to act as a cellular assembly line.

   b. CAPABLE*Assembly* methods at present do not cover the planning of mixed-model assembly lines. The system is incapable of performing parallel assembly planning and optimisation.

   c. Machine operation times have to be provided by the user. All other assembly operation and handling times are stored in operation-specific databases.

7.2.3 Functionality

It is the belief of the author that computer systems such as CAPABLE*Assembly* should be used to support decision-making rather than try to automate everything, since humans are still better in decision making than computer systems. The main objectives of CAPABLE*Assembly* with respect to concurrent engineering are generally aimed at:

1. Improving product quality; or the extent to which a product satisfies customer requirements. This has both objective and subjective attributes.
2. Reducing lead times; that is, reducing the time from product concept to successfully bringing the product to the market. This is often termed "time to market".
3. Reducing product cost; where cost can be defined as the level of resources required taking the product from concept to market. This includes the hours worked on the product, materials used in the product and any equipment or services that are used.

7.3 Verification of the various hypothesis and associated modelling methods

7.3.1 Assembly time evaluation

The following series of assembly tasks were devised to examine the accuracy of the resulting assembly time expressions generated by SPAM. These tasks will allow for assembly actions to be scrutinised, increasing the ability to better gauge the associated index value for basic assembly motion. It is hoped that body motions or slight variations in body motions that were previously not considered, or overlooked will be brought to light.

The assembly task devised to test the assembly time expressions derived using SPAM required the following characteristics to simulate an actual assembly situation;

1. The assembly task needed to be simple enough to produce learning in a short period of time (approximately 15 minutes) with initially naive operators.
2. The potential to check different assembly sequences.
3. The assembly task had to replicate real industrial operations.

In order to test SPAM two simple assembly tasks were designed. These involved the manual assembly of a pseudo-flange assembly and riveting of thin metal sheets in

simulation of real assembly task. The workplace layout and assembly tasks are described in the following sections.

7.3.1.1 Pseudo-Flange Assembly

Apparatus: The following equipments were used to perform this experiment.

- Two circular wooden discs with six pre-drilled holes
- Three part bins consisting of ten bolts (M10), ten washers (M10) and ten nuts (M10).
- One closed end spanner
- One engineer's vice
- One G-clamp
- Stopwatch

The parts for two pseudo-flange assemblies were laid out using an ergonomic design (Braun, Rebollar, and Schiller, 1996). All components were within the zone of convenient reach and in locations that allowed maximum use of simultaneous motions. The general layout of the workplace is shown in Figure 7-1. The construction of the jig for the pseudo-flange assembly is shown in Figure 7-2. The author demonstrated the desired method of assembly. The assembly sequences tested include codes BN_1, BN_2, BNW_4, and BNW_6, as described in Appendix E.

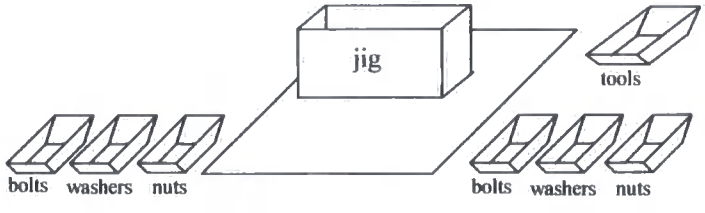


Figure 7-1: General layout of manual assembly workplace

For BN_1 these were; (1) Reach to the part bin containing bolts and collect a handful of bolts from the bolt bin with the right hand. (2) Insert bolt through holes of the pseudo-flange, repeat process until six bolts have been placed in position. Return remaining bolts to part bin if necessary, and this action should be a toss[1] action. (3) Reach for a single nut with the right hand from the nut bin, engage and rundown nut using the

[1] The toss action as defined by MOST system has a placement (P) parameter (see Chapter 2) index value of 0. A time penalty will not be incurred if the remaining parts are tossed into the respective part bins.

closed-end spanner to tighten the nut using a single wrist[2] action. Repeat process for each bolt.

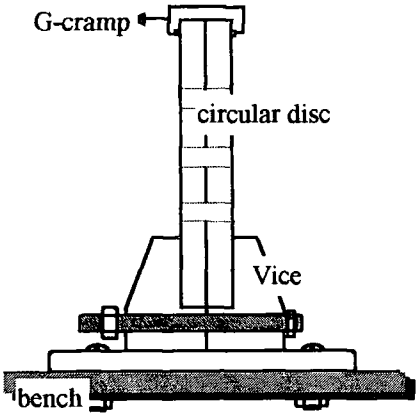


Figure 7-2: Jig set-up for pseudo-flange assembly

For the BN_2 (Bolts and Nuts sequence 2) sequence, these were; (1) reach to the bolt bin and collect a single bolt from a part bin using the right hand. (2) Insert bolt through holes on the pseudo-flange. (3) Reach for a single nut from the nut bin with the right hand. (3) Engage nut with bolt and rundown. (4) Tighten nut with a closed-end spanner using a single wrist action. (5) Repeat the entire process for each of the six holes on the flange.

For the BNW_4 (Bolts, Nuts and Washers sequence 4) sequence, these were; (1) simultaneously reach for the bolt and washer bins using both left and right hands to collect a single washer and bolt from their respective bins place washer on bolt. (2) Insert bolt through holes on the pseudo-flange repeat for all six holes on the pseudo-flange. (3) Get a nut from the nut bin engage and rundown nut. Tighten using a closed-end spanner with a single wrist action. Repeat process for all bolts.

For BNW_6 (Bolts, Nuts, and Washers sequence 6) sequence, these were; (1) simultaneously reach for a handful of bolts and washers from their respective part bins. (2) Place washer on bolt and insert bolt through holes on the pseudo-flange, repeat for all holes on pseudo-flange return all remaining parts to their respective bins using a toss action. (3) Get a nut from the nuts bin engage and rundown nut. Tighten using the closed-end spanner with a single wrist action. Repeat process for all bolts.

7.3.1.2 Pseudo-Riveting Assembly (PRA)

Apparatus: The following equipments was used to perform this experiment.

- Two sheets of metal with pre-drilled holes.

[2]The wrist is rotated through 90°.

- One part bin consisting of blind rivets.
- Riveting tool
- Vice / Clamping device

Two thin metal sheets were to be riveted together using a hand held riveting tool. The dimensions of the sheets of metal are thickness: 3mm, length 30mm and height 15mm. The sheets of metal have six holes drilled lengthwise and three holes breadthwise as shown in Figure 7-3. The holes diameters were 3.3 mm in accordance to the recommended clearance hole diameter from the product catalogue (RS Speedriv Riveting System).

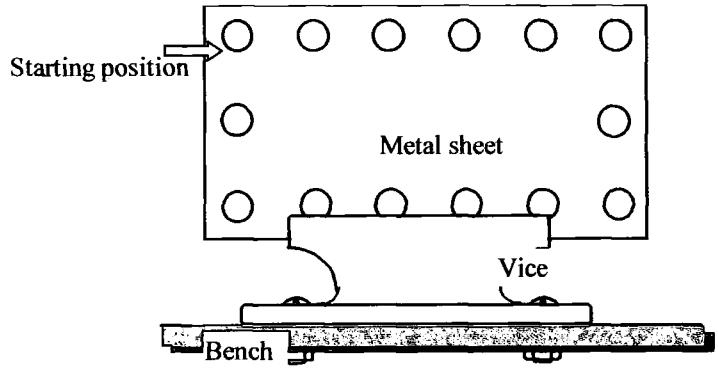


Figure 7-3: Jig set-up for riveting

The RS Speedriv Riveting System, which consists of aluminium alloy blind rivets, and a riveting tool (light duty manual tool for blind rivets), was used for the riveting process. The general layout of the workplace follows the layout shown in Figure 7-1. The construction of the jig for the PRS is shown in Figure 7-3. The author demonstrated the desired method of assembly. The assembly sequences tested include codes RIV_1 and RIV_2.

For the case of RIV_1 (Rivets sequence 1) assembly operation sequence, these were; (1) with the riveting tool in the right hand, reach to the rivet bin and collect a single rivet from the rivet bin using the left hand. (2) Insert the shank of the rivet into the nose of the riveting tool. (3) Insert the head of the rivet into the hole on the top left of the metal sheet as shown in Figure 7-3. (4) Actuate the riveting tool using one working stroke. (5) Repeat the process (1) to (4) for the six holes.

For the case of RIV_2 these were; (1) reach to the rivet bin and collect a single rivet from the rivet bin using the right hand. (2) Insert rivet through holes on metal sheet. (3) Reach for the riveting tool with the right hand. (4) Insert the rivet mandrel into the nose of the riveting tool. (5) Actuate the riveting tool. (6) Repeat process (1) to (5) for six holes of the metal sheets.

The BN_1, BN_2, BNW_4, BNW_6, RIV_1, and RIV_2 assembly operation sequences are typical of assembly operations found in high volume production industries, such as the automotive, aeronautical, household and outdoor appliances industries. The derivation and subsequent analysis of such sequences can aid the development and training of assembly workers, increase reliability by reducing mistakes and hence rejects on manual assembly lines. They also provide a more accurate means of measuring, predicting, and attaining production rates.

7.3.1.3 Operators

Three male operators assembled the pseudo-flange and pseudo riveting assemblies six times per session over a period of one day, three sessions per day. The operators were engineering students between the ages of 24 and 30 who volunteered to do the experiments. No operators had previous experience in assembly work. They all reported good health and had no physical disabilities, such as impaired vision or restricted mobility.

7.3.1.4 Procedure

The method of assembly was demonstrated twice and the operators were instructed to use the exact method demonstrated. The operators were informed that their learning and performance was to be recorded. The main instructions were to concentrate on the task and to work at maximum speed.

It was suggested that the operators were to be seated in an upright position and their chairs adjusted so that their elbows were 50-100mm above the working surface of the table. Operators were however given the opportunity to attain their most "natural" assembly position. It was found that all operators preferred to perform the assembly sequences in an upright standing position. Operators were allowed four practices run on each assembly before the start of each experiment. The operators were timed on the construction of two pseudo-flange/riveting assemblies. No rest periods were provided during the construction of the pseudo-flange or pseudo-riveting assemblies but a five-minute rest period was allowed between the completion of the pseudo-flange and/or riveting assembly and before the beginning of the next assembly.

7.3.1.5 Results

Pseudo-flange Assembly

The mean assembly times for each trial were calculated and are given in Table 7-1. Each operator had six trials at each assembly task; a mean time for each trial was then

calculated. An average for each assembly task is subsequently generated, and these results are also shown in Table 7-1.

	Mean Assembly Times (seconds)						
Ass Task Code	Mean time for Trial 1	Mean time for Trial 2	Mean time for Trial 3	Mean time for Trial 4	Mean time for Trial 5	Mean time for Trial 6	Average time
Bn_1	37	34	32	31	28	26	31
Bn_2	48	47	44	42	40	40	44
Bn_3	40	37	37	36	34	33	36
Bnw_4	41	40	39	38	36	34	38
Bnw_6	40	40	39	35	32	37	37

Table 7-1: Mean assembly times for pseudo flange assembly

Pseudo-Riveting Assembly

The mean assembly times for each trial were calculated and are given in Table 7-2. Each operator had six trials at each assembly task; a mean time for each trail was then calculated. An average time for each assembly task is subsequently generated these results are also shown in Table 7-2.

	Mean Assembly Times						
Ass Task Code	Mean time for Trial 1	Mean time for Trial 2	Mean time for Trial 3	Mean time for Trial 4	Mean time for Trial 5	Mean time for Trial 6	Av. Time
Riv_1	29	28	28	26	24	24	25
Riv_2	36	34	33	32	30	30	33

Table 7-2: Mean times for pseudo-riveting assembly

The maximum percentage deviation in the results was found to be approximately eight percent. There appears to be a systematic offset between calculated times and actual assembly task times as shown in Figure7-4. Figure 7-4 also shows a comparison between the assembly times calculated using SPAM, the actual assembly times, and MOST systems. The times calculated using MOST were obtained using methods as described in Section 2.9.4.

7.3.1.6 Analysis of results

On analysis, the results of both experiments shows there is a constant difference (see Figure 7-4) between the experimental assembly times and the times obtained using SPAM, of ±5%. This margin is estimated, taking into consideration the operator used in this experiment are not skilled operators. It is already known from pervious analysis of predetermined motion times systems (Gendaidy et al, 1990; Kanai et al.) that assembly times obtained using pre-determined motion times have an inherent error margin of approximately 5% when compared to times obtained in industry. Industrial times tend to be significantly higher which is consistent with the results presented. Therefore, it is to be expected that assembly task times obtained using SPAM would yield magnitude similar to MOST systems. The results show a higher degree of consistency in SPAM

compared to MOST systems relative to the actual assembly times. This is due to the classification of index parameters within MOST systems. As the size of objects is not dealt with in as much detail, and the orientation of parts is less accurately defined, the scope for errors at the boundaries of each given class is more evident.

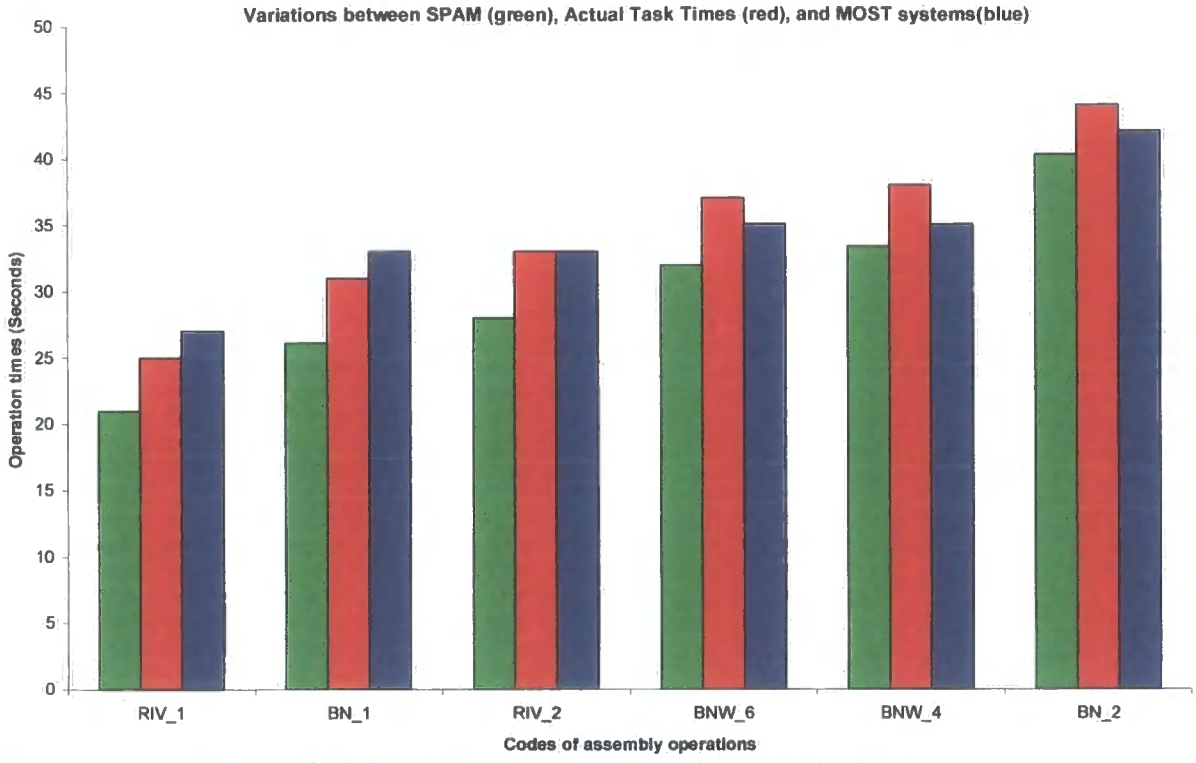


Figure 7-4: Comparison of SPAM, MOST, and Actual assembly operation times

The difference incurred is largely due to the assumptions made to cater for lack of information and level of accuracy desired. These assumptions include;

1. All parts are within arms reach that is the arc of an out stretched arm.
2. Parts present no handling difficulties, such as in slippery surfaces or parts less than 2mm in thickness and 15 mm in size.
3. For the BN and BNW assembly sequences, the nut is initially rundown the shaft of the bolt. The spanner is only required for tightening purposes. Hence, the number of revolutions requires to securely fasten the bolt using a spanner is one.
4. The order to which the bolts are inserted into the holes is irrelevant, as each hole on the disc is situated less than two inches apart, and is equidistant.
5. All assemblers are not skilled workers in the associated experimental fields.
6. For fastening purposes, the action of a manual lever rivet gun is mimicked as that of clippers.

7. After each sequence has been performed, the holes need to be drilled out. The effect of this was deemed insignificant.

Initially, some assembly task times were considerably higher than that of the calculated times. These inconsistencies were only prominent in the assembly task that involved the use of multiple part handling and fastening motions involving an arm motion, i.e. riveting.

In situations where a handful of parts were obtained from a part bin by the assembler for both bolting and riveting, a significantly lower assembly time was noticed. The operators subsequently performed these assembly tasks again. The emphasis here was on the allocation of index parameter values for body motions. As a result a "palming action" was introduced (see section 4.6.4).

However, discrepancies in riveting times were still prominent in addition to the changes explained above. Again, the relevant task had to be performed again. It was noticed that for the case of a thin metal sheet, the assembler required more than one working stroke and applied a greater force when riveting. Parameter index values were subsequently calculated to allow for "thin metal" sheets.

The number of riveting sequences used for the purpose of these validations is significantly less than that used for bolting sequences. This is due to tool availability. The majority of sequences derived are suitable for other hand held tools such as the air-riveting gun. Sequences as code RIV_5, which is shown in Appendix E, proved impossible to execute using a lever riveter.

It was hoped to achieve a fairly constant deviation between assembly task times and calculated times. The calculated times were expected to be lower (maximum of ±5%) than the measured times as this would be consistent with other findings. The results shown in Figure 7-5, demonstrate that there is a consistent difference between actual assembly times and times calculated using SPAM of around 5%.

A more complex and extended assembly process could be designed to observe body motions in more detail and in varying situations. In this study, the significant learning period was short and the analysis was therefore limited. This period should also be extended, as PMTS do not attempt to model novice and transition to an expert.

Based on the results of the experiments described above the following conclusions can be drawn:

1. SPAM can be used to derive expressions and/or values for the calculation of assembly times for standard assembly operations.
2. Assembly times obtained using SPAM have an expected maximum deviation of approximately ±4%.
3. Assembly operations with higher TMU (time measured units, as defined in MOST systems) values generally have a higher degree of accuracy in results obtained.
4. These experiments have served to prove the validity of SPAM. More experiments are required to fully validate the method.

Errors in the initial codification of basic assembly motions have been noted and rectifications have been made where deemed necessary. Various synthetics were subsequently derived for multiple tool use and part handling (see Appendix E).

7.4 Industrial case studies

The industrial case study was undertaken in collaboration with a well-known manufacturer of lightweight garden tools. The four conceptual designs were constructed with the aid of the industrial designer in the manufacturers research and design department. A foam model of the product was used to obtain external dimensions and assembly features. All other part data (dimensions, assembly features, part weight), with regards to internal components such as switches, wires, and motors, were obtained from previous designs of the low cost trimmer. In-house jigs/fixtures were also modelled. The standard part database was also updated to reflect components regarded as standard parts within this industry; typical custom standard parts include motors, and switches.

The naming convention for the product models uses capitalisation to differentiate between variables. For example, TwoShellMiniTrim_screws identifies a product whose main body is made of two main body moulded shells that are screwed together prior to product testing.

The aim of the case study is to generate optimal assembly plans for four conceptual designs of a low cost trimmer, given the manufacturer's current factory layout. The four conceptual designs use different assembly processes to assemble the trimmer. As the final product design will have to be designed with specific assembly processes in mind (the design of the moulds of the main body shells will depend on whether the shells are to be welded together or screwed), it is desirable to have comparative data on the possible effects of the different assembly processes used to assemble the product on assembly time, and thus cost.

A detailed explanation of the generation of the assembly plans for one product model (TwoShellMiniTrim_welding) is initially presented, showing the routes taken from the product model to the generation of an optimal assembly plan using the hypothesis and methods discussed in the preceding Chapters.

Subsequently, the assembly plans for the remaining three conceptual designs are presented, followed by a discussion and the conclusions that can be drawn as a result of the analysis.

7.4.1 Product model 1: TwoShellMiniTrim_welding

TwoShellMiniTrim_welding describes a low cost trimmer where the main body is formed by joining two main body moulded shells. Once all components have been assembled the main body shells are welded together before testing and packaging processes are performed. The product model is shown in Figure 7-5. Distinguishing product features of interest include:

1. Welding features
2. 2 main body shells
3. Pressure fit connection between mains wire and switch

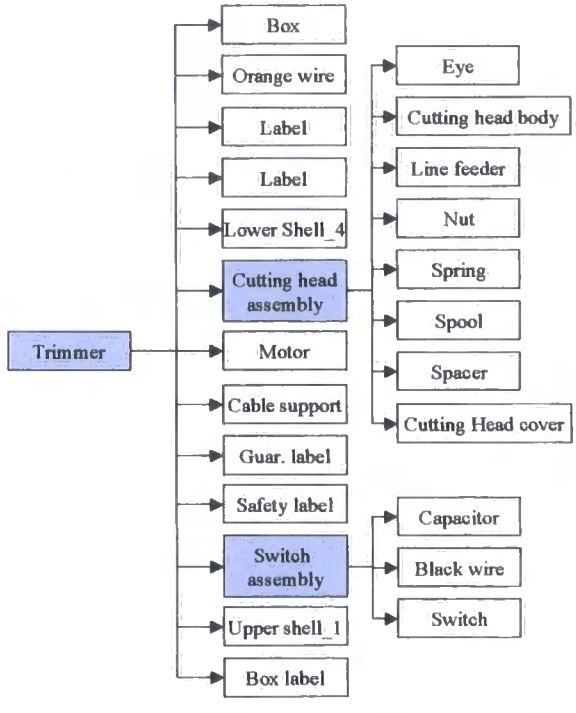


Figure 7-5: Product model for TwoShellMiniTrim_screws

An aggregate product model is constructed by extracting relevant assembly features such as blind hole, v slots, external and internal threads, from each component of the product model TwoShellMiniTrim_welding. The structure of the product model is

imposed on the aggregate product model. The components of the conceptual mini trim are made-up from existing models of trimmers currently in production. The dimensions for all the components shown in Figure 7-5 were obtained from engineering drawings of the existing components. Once the dimension of the component is obtained, an estimate of the component's weight is calculated based on its positive feature. For complex bought-in subassemblies, such as the motors used, the component weight is obtained using weighing scales.

The assembly features, both positive and negative, of individual components are linked, emulating assembly operations. Within CAPABLE*Assembly*, this process creates an AFC. Each AFC contains the following inherent data:

1. Assembly operation type; examples covered include adhesive, thermoplastic welding, threaded, placement, pressure-fits packaging, and riveting assembly operations.
2. Mating components. These are the components being assembled.
3. Assembly operation time. The assembly operation times stored in the AFC are obtained from operation-specific databases, Boothroyd and Dewhurst part handling times, and, where possible, using SPAM, via the assembly time generation algorithm AGA, see Section 4.4.
4. Floating AFC. This field is set in the generation of the connectivity model. It is stored as a logical. It determines if the AFC is free to move between/within assembly levels.
5. Fixed AFC. Also stored as a logical and set during the course of generating the connectivity model. It determines if the AFC is restricted to its current assembly level, and fixed with respect to its neighbouring AFCs.
6. Assembly operation precedence rating. The precedence rating is set using the precedence constraints algorithm (Section 4.3.2). The type of AFC determines the precedence rating of an AFC. Generally, permanent AFCs have higher precedence rating compared to reversible AFCs (see Table 7-2).
7. Assembly tool type required. Stores the tool or machine required to perform the AFC, if any.

The aggregate product model is used to create the connectivity model shown in Figure 7-6, hence the connectivity model inherits its structure from the product model. The

connectivity model is generated using the contact, precedence and technological constraints algorithms. These algorithms are used to ensure the feasibility of the assembly sequences obtained from the aggregate product model. It [connectivity model] includes restrictions as to when assembly operations can be performed. The AFCs shown in green represent fixed AFCs, which are assembly operations restricted to a given assembly level. As a result of the precedence, contact, and technological constraints used for the generation of the connectivity mode, any assembly sequence generated from the connectivity is a feasible assembly sequence.

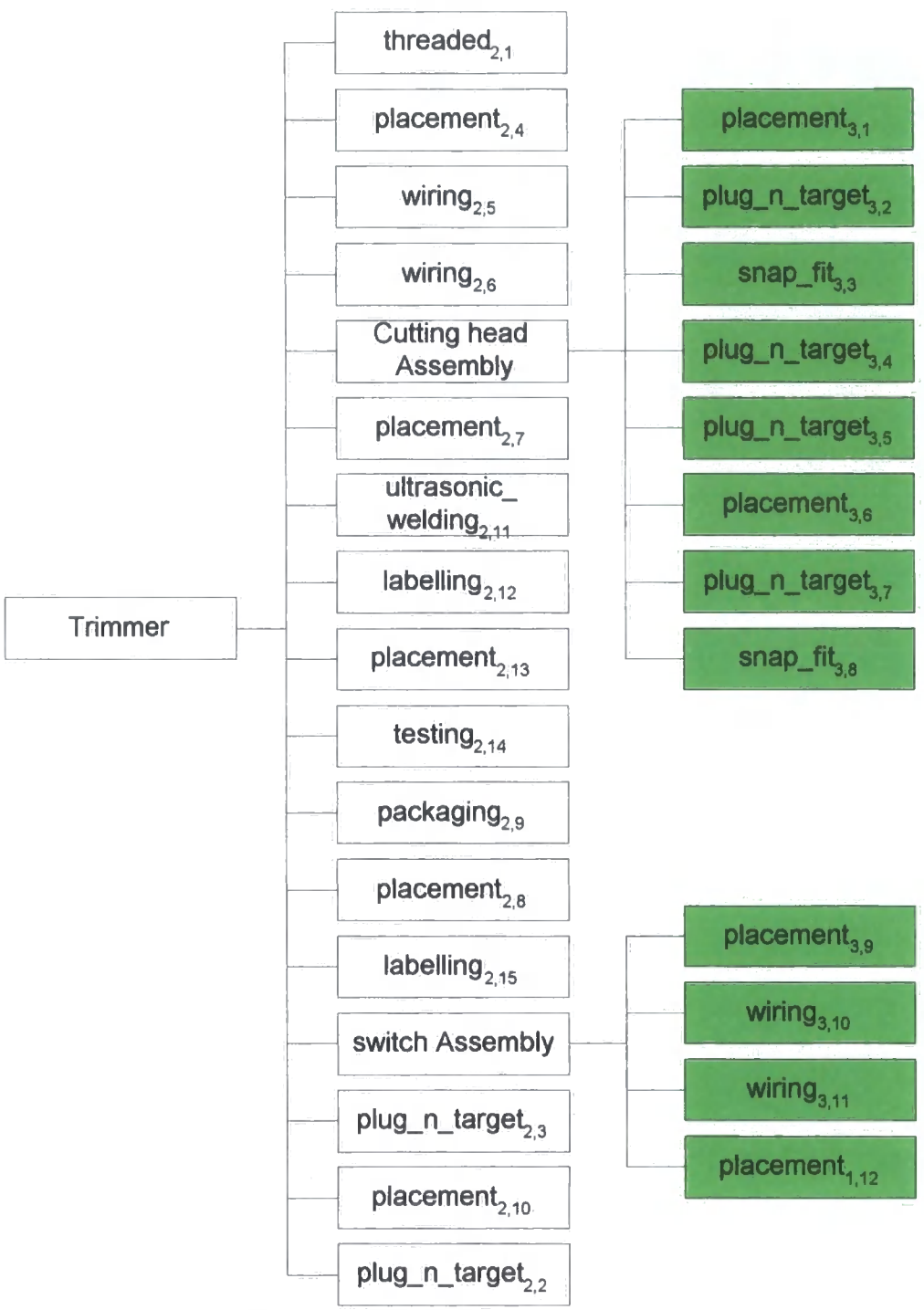


Figure 7-6: Connectivity model of TwoShellMiniTrim_welding.

The initial assembly sequence derived from the connectivity model is shown in Table 7-3. The initial assembly sequence is obtained using a simple bottom-top assembly approach. The algorithm starts with the largest part in the highest assembly level and assembles the trimmer based on the introduction of the mating parts in successive AFCs. The mating parts of each AFC are also shown in Table 7-3.

Initial assembly sequence	AFC type	Mating Parts
1	$placement_{3,1}$	stand_alone,cutting_head_base
2	$plug\_n\_target_{3,2}$	cutting_head_base,nut
3	$snap\_fit_{3,3}$	cutting_head_base,eye
4	$plug\_n\_target_{3,4}$	cutting_head_base,spacer
5	$plug\_n\_target_{3,5}$	spring,line_feeder
6	$placement_{3,6}$	cutting_head_base,line_feeder
7	$plug\_n\_target_{3,7}$	cutting_head_base,spool
8	$snap\_fit_{3,8}$	cutting_head_base,cutting_head_cover
9	$placement_{3,9}$	fixture,lower_shell
10	$wiring_{3,10}$	flymo_switch,flymo_capacitor953506932
11	$wiring_{3,11}$	flymo_switch,black_wire
12	$placement_{3,12}$	lower_shell,switch_ass
13	$wiring_{2,1}$	switch_ass,orange_plug_wire
14	$wiring_{2,2}$	flymo_motor953491634,switch_ass
15	$threaded_{2,3}$	flymo_motor953491634,cutting_head_ass
16	$plug\_n\_target_{2,4}$	cable_support,orange_plug_wire
17	$placement_{2,5}$	lower_shell,flymo_motor953491634
18	$plug\_n\_target_{2,6}$	lower_shell,cable_support
19	$placement_{2,7}$	upper_shell,lower_shell
23	$testing_{2,8}$	test_mc,lower_shell
20	$ultrasonic\_welding_{2,9}$	upper_shell,lower_shell
21	$labelling_{2,10}$	upper_shell,flymo_label
22	$placement_{2,11}$	parts_pack,gtee_label
24	$packaging_{2,12}$	box, lower_shell
25	$labelling_{2,13}$	upper_shell,serial_label
26	$placement_{2,14}$	box,guard
27	$placement_{2,15}$	box,parts_pack

Table 7-3: Initial assembly sequence

The initial assembly sequence starts by assembling the cutting head assembly ($placement_{3,1}$). The main body of the cutting head is first placed in a stand-alone fixture; all other components in contact with the cutting head main body are then sequentially assembled. The switch assembly is subsequently assembled.

Next, the lower shell is placed onto a fixture ($placement_{3,9}$), which is fitted to a conveyor belt controlled by the operator at the designated workstation. Theoretically (and in practice), this AFC can occur before the cutting head assembly is assembled. However, this AFC ($placement_{3,9}$) occurs as late as possible within level 3 because:

- There are more components in contact with the cutting head assembly within level 3.

- The cutting head does not have any direct links to AFCs in assembly level 2.
- The lower shell is constrained (directly linked) to an AFC in level 2 ($placement_{2,5}$), a lower assembly level.

The information with regards to components in contact with each other, and their relation in terms of ordering AFC is obtained from the contact and precedence constraint algorithms.

The switch assembly is then attached to the motor and the remaining components attached to the main body shell of the trimmer. The top shell is placed on the lower shell and the whole assembly is tested to ensure the product meets all functional and quality requirements before the two halves are ultrasonic welded together. The assembled product is packed and the assembly process is complete.

The initial assembly sequence is encoded as discussed in Section 5.8.2; this is used as the input to the simulated annealing optimisation process. The simulated annealing algorithm attempts to find an optimal assembly sequence based on minimum assembly time (c) by minimising the number of reorientations of mating components (Section 5.5.1), maximising parallelism (Section 5.5.2), and maximising the stability of intermediate subassemblies (Section 5.5.3). Equation 5-8 gives the overall expression for minimising assembly time. The weightings for all the assembly variables in this formula are set to 1.

$$c = (w_{re})\frac{X_{re}}{c_1} + (w_{pa})^{*} c_2 + (w_{st})\frac{c_3}{X_{st}} \qquad \textbf{Equation 5-8}$$

For the purpose of this analysis the following simulated annealing parameters were used to generate the globally optimised assembly sequence:

1. Initial temperature of the system is 100°C.
2. Maximum number of cooling schedules = 5.
3. A cooling rate of 0.95.
4. The ratio of accepted solutions to the number of solutions generated is 0.9. This is the termination criterion. In other words, the algorithm will terminate when it starts accepting almost every solution it randomly generated, indicating an optima has been found.

The result of the simulated annealing algorithm has been truncated in Figure 7-7. This is to show the behaviour of the system at the beginning of the optimisation process.

Although the number of AFCs is large the algorithm appears to find good solutions relatively quickly. It was expected that a larger number of solutions would be visited before the solution began to converge, due to the number of AFCs considered.

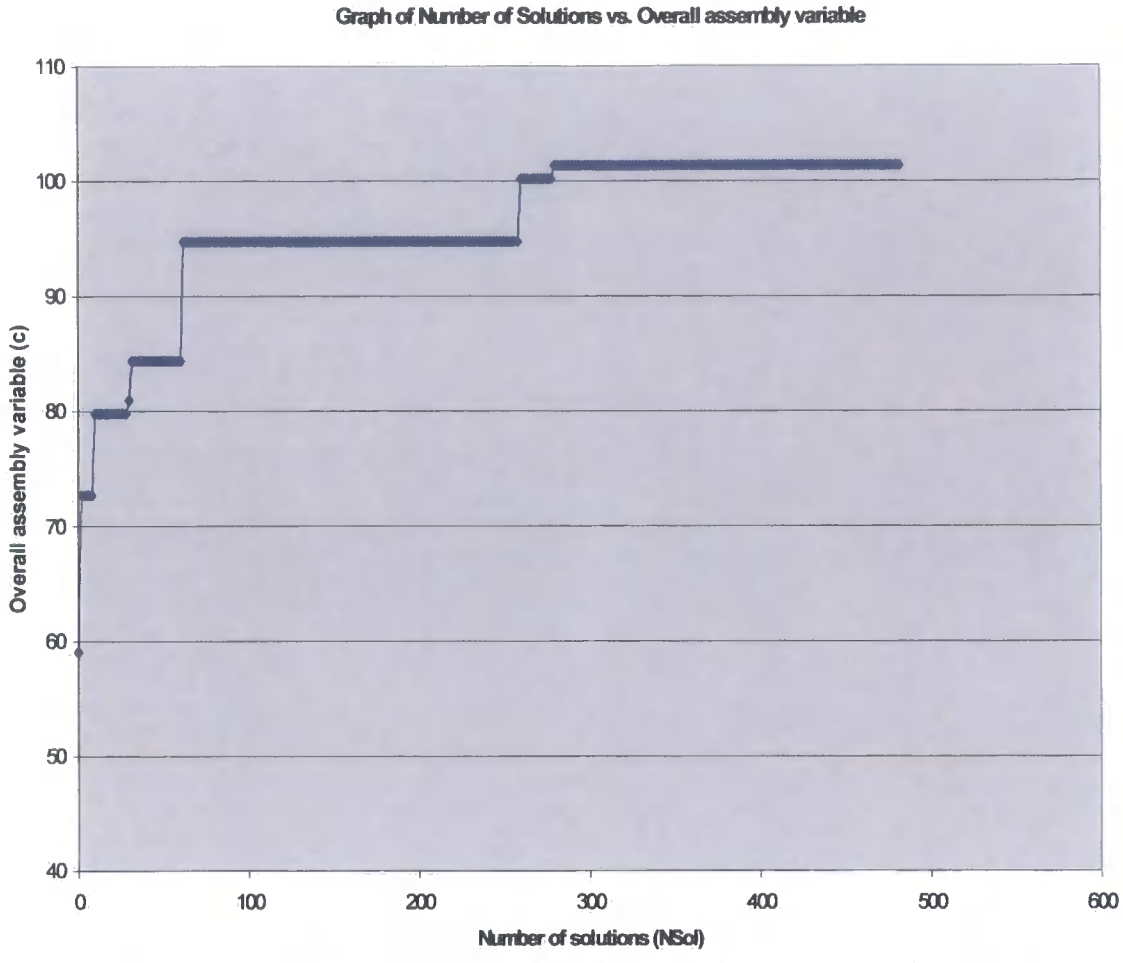


Figure 7-7: Minimisation of assembly time; Graph of number of solutions visited vs. overall assembly variable

Furthermore, the optimisation process was completed in 38 seconds. Based on previous analysis, a CPU time of at least 80 seconds was expected in accordance with smaller preliminary trials. The reason for this can be associated to the number of fixed AFCs in the assembly sequence (11 of the 27 AFCs in the assembly sequence are fixed), limiting the number of solutions that would be deemed feasible and thus evaluated. Such a large number of AFCs could be classified as fixed due to large amount of in-house knowledge of the product design available at the onset of this case study. This serves to prove how useful such a system could be to a design engineer as such data would be readily available.

The system does display a tendency to get stuck in local optima, shown by the steps in Figure 7-6. Whilst the system is able to eventually start finding better solutions, it does leave open the question if the true optimum solution has been found. Whether or not the true optimum has been found is irrelevant, the aim here is to generate an optimal

solution, of which there are a few. Convergence within the simulated annealing algorithm is achieved when the majority of the new solutions evaluated are not rejected by the algorithm (acceptance ratio = 0.1). If the acceptance ratio is decrease the simulated annealing algorithm continues (sooner or later) to find better solutions. However, the improvement in the sequences generated offer little benefit, especially when compared with the CPU time sacrificed; takes an average of 32 seconds to improve the overall assembly variable by approximately 1.6 units, a total of 204 solutions were evaluated before a better solution was found, as shown in Table 7-4.

Number of solutions evaluated (NSol)	Overall assembly variable (c)
278	100.175
280	101.382
282	101.382
480	101.382
482	101.382
484	102.95

Table 7-4: Extract of simulated annealing results

Optimal assembly sequence	AFC type	Mating Parts
1	$placement_{3,1}$	stand_alone,cutting_head_base
2	$plug\_n\_target_{3,2}$	cutting_head_base,nut
3	$snap\_fit_{3,3}$	cutting_head_base,eye
4	$plug\_n\_target_{3,4}$	cutting_head_base,spacer
5	$plug\_n\_target_{3,5}$	spring,line_feeder
6	$placement_{3,6}$	cutting_head_base,line_feeder
7	$plug\_n\_target_{3,7}$	cutting_head_base,spool
8	$snap\_fit_{3,8}$	cutting_head_base,cutting_head_cover
9	$threaded_{2,3}$	flymo_motor953491634,cutting_head_ass
10	$placement_{3,9}$	fixture,lower_shell
11	$plug\_n\_target_{2,4}$	cable_support,orange_plug_wire
12	$plug\_n\_target_{2,6}$	lower_shell,cable_support
13	$wiring_{3,10}$	flymo_switch,flymo_capacitor953506932
14	$wiring_{3,11}$	flymo_switch,black_wire
15	$wiring_{2,2}$	flymo_motor953491634,switch_ass
16	$placement_{3,12}$	lower_shell,switch_ass
18	$wiring_{2,1}$	switch_ass,orange_plug_wire
17	$placement_{2,5}$	lower_shell,flymo_motor953491634
19	$placement_{2,7}$	upper_shell,lower_shell
20	$testing_{2,8}$	test_mc,lower_shell
21	$ultrasonic\_welding_{2,9}$	upper_shell,lower_shell
22	$labelling_{2,10}$	upper_shell,flymo_label
23	$placement_{2,11}$	parts_pack,gtee_label
24	$packaging_{2,12}$	box,guard
25	$labelling_{2,13}$	upper_shell,serial_label
26	$placement_{2,14}$	box,lower_shell
27	$placement_{2,15}$	box,parts_pack

Table 7-5: Optimal assembly sequence

The optimal assembly sequence, shown in Table 7-5, initially follows the same path as the initial assembly sequence. The cutting head is assembled using exactly the same assembly sequence, followed by the switch assembly. This pattern is not surprising considering all the AFCs ($placement_{3,1} \rightarrow placement_{3,12}$) in level 3 are fixed AFCs.

The assembly sequence in level 3 has been altered, with three AFCs ($plug\_n\_target_{2,4}$, $plug\_n\_target_{2,5}$, and $wiring_{2,2}$), from level 2 being assembled in level 3 (shown in red in Table 7-5). The cable support used to secure the mains wire (orange_plug_wire) to the main body shell, is now assembled in level 3. This is done prior to attaching the mains wire to the switch assembly ($wiring_{2,1}$), which is still performed in level 2. This move increases the stability of the assembly. Also, the motor is now attached to the switch assembly ($wiring_{2,2}$) before the two parts are fixed to the lower shell ($placement_{3,12}$ and $placement_{2,5}$). This move is due to minimising reorientation of mating components, rather than stability. The assembly would be just as stable (and perhaps more stable) if the two components were secured to the lower shell, before they were assembled.

The optimal assembly sequence is used as the input to genetic algorithm. The plan of assembly line provided by the manufacturers that will be used to assemble the low cost trimmer is shown in Figure 7-9. As the limitations of the assembly line is known, a brown field assembly line is generated. The assembly line is modelled by creating the workstation objects, at runtime (as described in Section 6.6). The available resources associated with each workstation such as tool availability and labour, are stored in databases. Such information is obtained and added to the workstation object when a workstation object is created. Workstation numberings (Wks(*number*)) start from 0 to 3, for four workstations. The assembly starts at Wks*0* and is completed in Wks*3*. Each workstation has the following properties:

1. Workstation number. The workstation number is used to identify a given workstation.
2. Workstation cycle time. The workstation cycle time is calculated when generating optimal assembly plans as part of the assembly line balancing module.
3. Workstation idle time. This difference between the total assembly time of assembly operations loaded on a workstation, and the workstation cycle time.
4. Assembly tool types. This refers to the assembly tools and/or machines loaded on a given workstation.

5. Workstation assembly task. Holds all assembly operations assigned to the workstation.

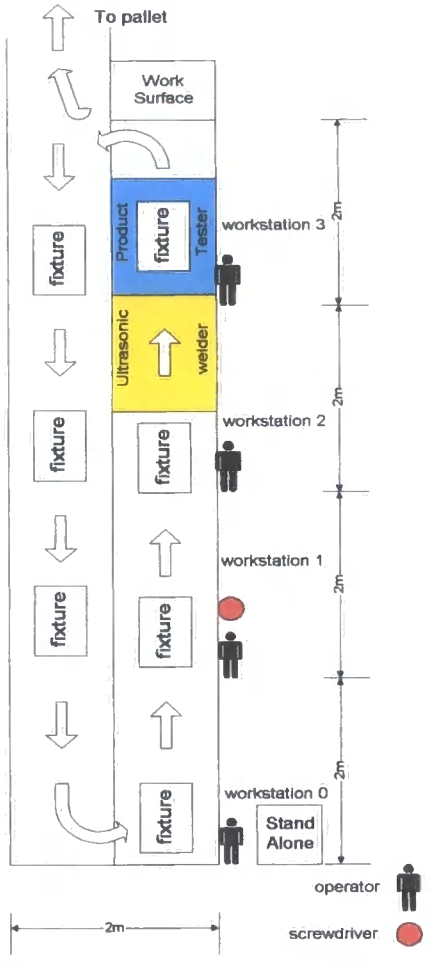


Figure 7-8: Assembly line layout for TwoShellMiniTrim_welding

For the purpose of this analysis the following genetic algorithm parameters were used to generate an optimal assembly sequence.

1. Population size is 110
2. Number of generations is 1000
3. The convergence criterion is based on the number of generations.
4. Genome length is 27
5. Crossover rate s 0.6
6. Mutation rate is 0.3

Table 7-6 shows the assembly data used to generate the optimal assembly sequence based on minimising cycles time. The number of workstations used is as specified by the manufactures.

TwoShellMiniTrim_welding	
Factory loaded	Pseudo_Mini_Trim
Total assembly time	121.47 seconds
Production rate	1500
Production rate per	Day
Number of shifts	2
Length of shifts	8hrs
Maximum workstation cycle time	38 seconds
Number of workstations loaded	4
Number of operators	4

Table 7-6: Assembly line balancing data for TwoShellMiniTrim_welding

The genome length is set to 27, generating a genome with 27 genes, each representing an AFC object. The position of the AFC within the genome is set to correspond to its position in the optimal assembly sequence. The genetic algorithm never alters this position. An allele set containing all four workstations is generated and randomly loaded on each gene (AFC object). This generates the initial assembly plan used to create the initial population.

As discussed in Section 6.8, sigma truncation is used to calculate the fitness of each genome in a population. Deterministic sampling is used to determine which genomes within a population are passed on to the next generation.

The procedure for generating optimal assembly plans based on the minimisation of cycle time is shown in Figure 6-15 (Section 6.8). The fitness score and objective score of each individual in a population is calculated using Equation 7-1 (see Section 6.8.4), and 7-2 (see Section 6.9) respectively.

$$f = obj\_score - (obj\_ave - c * obj\_dev) \qquad \textbf{Equation 7-1}$$

$$obj\_score = \frac{100}{d} + \frac{1}{\sigma} + \frac{\delta}{\sigma} + \frac{f(cycle\_time)}{\sigma} + \frac{f(precedence)}{\sigma} \qquad \textbf{Equation 7-2}$$

The result of the genetic algorithm is shown in Figure 7-10. The mapping of the genetic algorithm result to the optimal assembly sequence is shown in Table 7-7.

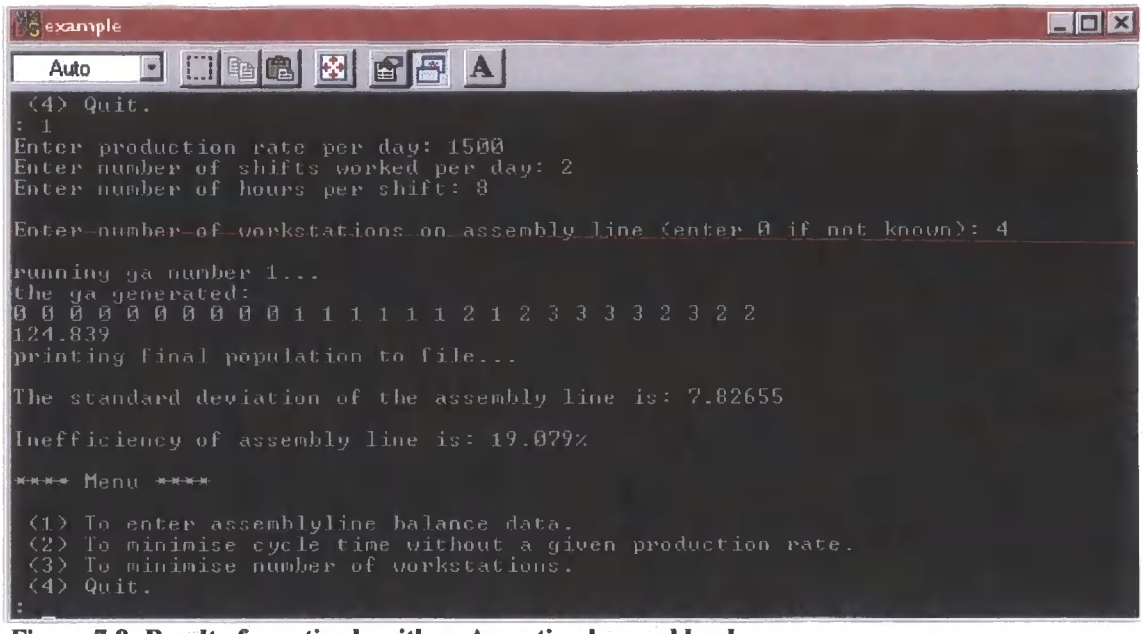


Figure 7-9: Result of genetic algorithm; An optimal assembly plan

Optimal assembly sequence	AFC type	Mating Parts	Workstation loading (Wks)
1	$placement_{3,1}$	stand_alone,cutting_head_base	Wks0
2	$plug\_n\_target_{3,2}$	cutting_head_base,nut	Wks0
3	$snap\_fit_{3,3}$	cutting_head_base,eye	Wks0
4	$plug\_n\_target_{3,4}$	cutting_head_base,spacer	Wks0
5	$plug\_n\_target_{3,5}$	spring,line_feeder	Wks0
6	$placement_{3,6}$	cutting_head_base,line_feeder	Wks0
7	$plug\_n\_target_{3,7}$	cutting_head_base,spool	Wks0
8	$snap\_fit_{3,8}$	cutting_head_base,cutting_head_cover	Wks0
9	$threaded_{2,3}$	flymo_motor953491634,cutting_head_ass	Wks0
10	$placement_{3,9}$	fixture,lower_shell	Wks0
11	$plug\_n\_target_{2,4}$	cable_support,orange_plug_wire	Wks1
12	$plug\_n\_target_{2,6}$	lower_shell,cable_support	Wks1
13	$wiring_{3,10}$	flymo_switch,flymo_capacitor953506932	Wks1
14	$wiring_{3,11}$	flymo_switch,black_wire	Wks1
15	$wiring_{2,2}$	flymo_motor953491634,switch_ass	Wks1
16	$placement_{3,12}$	lower_shell,switch_ass	Wks1
18	$wiring_{2,1}$	switch_ass,orange_plug_wire	Wks2
17	$placement_{2,5}$	lower_shell,flymo_motor953491634	Wks1
19	$placement_{2,7}$	upper_shell,lower_shell	Wks2
20	$ultrasonic\_welding_{2,9}$	upper_shell,lower_shell	Wks3
20	$testing_{2,8}$	test_mc,lower_shell	Wks3
21	$labelling_{2,10}$	upper_shell,flymo_label	Wks3
23	$placement_{2,11}$	parts_pack,gtee_label	Wks3
24	$packaging_{2,12}$	box,guard	Wks2
25	$placement_{2,14}$	box,lower_shell	Wks3
26	$labelling_{2,13}$	upper_shell,serial_label	Wks2
27	$placement_{2,15}$	box,parts_pack	Wks2

Table 7-7: Mapping of genetic algorithm result to optimal assembly sequence

As the position of the AFCs within the genome never changes, the result presented in Figure 7-10 can easily be decoded; each AFC is loaded on the workstation number currently occupying its position within the genome, as shown in Table 7-6. The optimal

assembly plan generated has an Objective score of 124.839 (shown in Figure 7-10), the final population for the genetic algorithm is shown in Figure 7-11. As in the illustrative example presented in Chapter 6, the objective score of individuals in the final population has a wide range (from 88.373 to 124.839), as shown in Figure 7-11, with only a few individuals having an objective score over 100. This can be controlled by the population size, as it limits the number of individuals used in creating subsequent generations. However, the large population size does ensure a wider range of results.

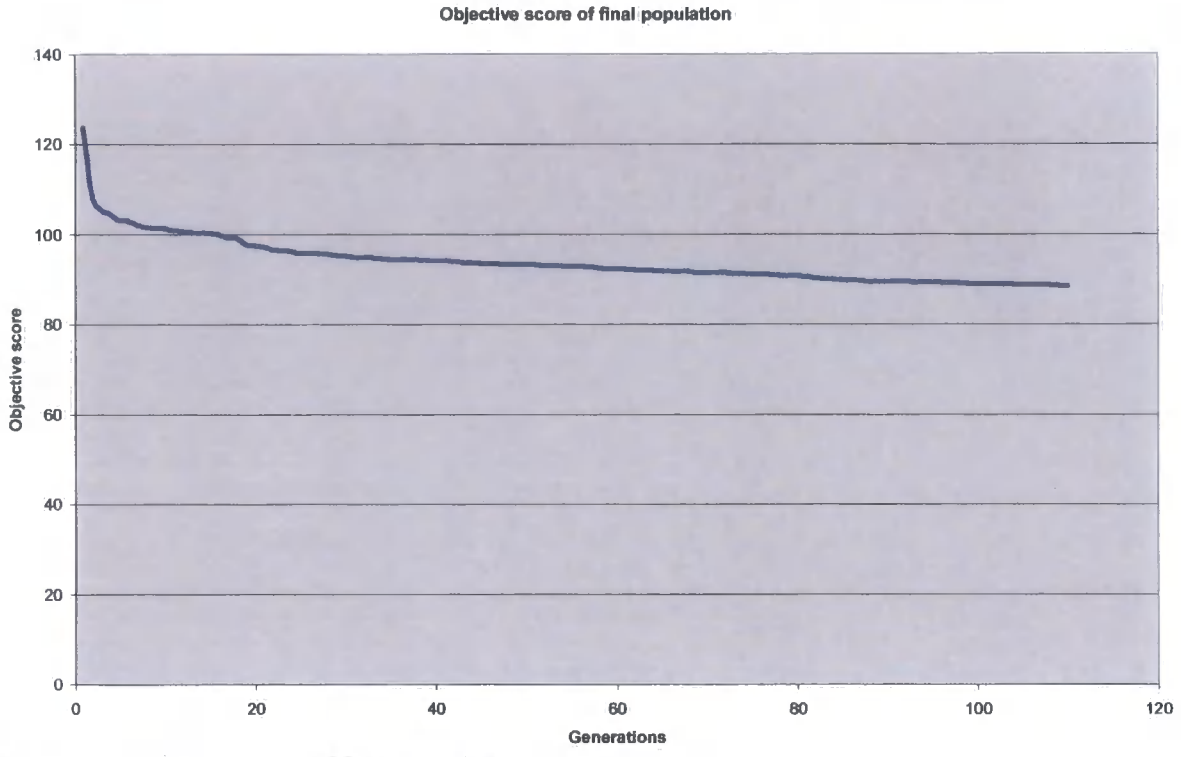


Figure 7-10: Objective score of final population

The station variation index (given by Equation 6-9) for the optimised assembly plan is 7.84 seconds. The station variation index is a measure of the degree of variation in total assembly operation time per workstation. A low value, as in this case, shows the workload has been evenly distributed between workstations, satisfying the criterion of workload smoothness. The balance delay of the assembly plan generated is 19.079%. The balance delay is a measure of the idle time on each workstation of an assembly line. This indicates that whilst the assembly line is efficient (~81%), there is still scope for improving the efficiency of the assembly line. Based on the balance delay and the station variation index, the manufacturers can afford to decrease the cycle time by increasing the production rate if desired. The issue of work-relatedness has also been addressed. Where possible the similar assembly operations and/or mating components have been loaded on the workstation ($wiring_{3,10}$, $wiring_{3,11}$,and $wiring_{2,2}$).

The loading of assembly operations on workstations is restricted by the feasibility checks (see Section 6.8.6) performed by the genetic algorithm. The checks ensure the appropriate tool in available for a workstation is to be assigned to an AFC. As a result, the assembly operation requiring the stand-alone unit ($placement_{3,1}$) is loaded on Wks*0*. Similarly, the process of welding the two main body shells ($ultrasonic\_welding_{2,9}$) is assigned to Wks*3*. The feasibility checks also ensure all AFCs allocated to a workstation can be performed within the given/evaluated workstation cycle. The maximum workstation time for this assembly plan is 33.84 seconds Wks*1*.

The genetic algorithm takes slightly longer to converge (CPU time of 91 seconds) when compared to the simulated annealing algorithm. This time can be reduced by decreasing the number of generations or by using the value of the objective score as the bases for convergence. Here, the genetic algorithm will converge if successive objective scores are within a predefined percentage. This method is susceptible to artificial convergence.

A detailed assembly plan is presented below.

Wks*0*

The assembly sequence for the operations loaded on workstation 0 is shown in Table 7-8. Total workstation assembly operation time – 32.40 sec

Available resources: Operator, 1 stand-alone unit

Identification code	AFC type	Mating parts	Operation time (sec)
connection953747873	placement	stand_alone,cutting_head_base	2.63
connection953747924	plug_n_target	cutting_head_base,nut	3.23
connection953747895	snap_fit	cutting_head_base,eye	3.83
connection953747945	plug_n_target	cutting_head_base,spacer	3.23
connection953747972	plug_n_target	spring,line_feeder	2.93
connection953748038	placement	cutting_head_base,line_feeder	3.45
connection953748065	plug_n_target	cutting_head_base,spool	2.93
connection953748130	snap_fit	cutting_head_base,cutting_head_cover	3.01
connection953748230	threaded	flymo_motor953491634,cutting_head_ass	3.71
connection953747763	placement	fixture,lower_shell	3.45

Table 7-8: Assembly operation loaded on Wks*0*

Wks*1*

The assembly sequence for the operations loaded on workstation 1 is shown in Table 7-9. Total workstation assembly operation time – 33.84 sec

Available resources: Operator

Identification code	AFC type	Mating parts	Operation time (sec)
connection953748475	plug_n_target	cable_support,orange_plug_wire	3.23
connection953748424	plug_n_target	lower_shell,cable_support	3.23
connection953747844	wiring	flymo_switch,flymo_capacitor953506932	4.35
connection953747810	wiring	flymo_switch,black_wire	3.53
connection953748376	wiring	flymo_motor953491634,switch_ass	5.55
connection953748348	placement	lower_shell,switch_ass	3.45
connection953748293	placement	lower_shell,flymo_motor953491634	10.5

Table 7-9: Assembly operation loaded on Wks*1*

Wks*2*

The assembly sequence for the operations loaded on workstation 2 is shown on Table 7-10. Total workstation assembly operation time – 32.36 sec

Available resources: Operator,

Identification code	AFC type	Mating parts	Operation time (sec)
connection953747633	wiring	switch_ass,orange_plug_wire	3.53
connection953748523	placement	upper_shell,lower_shell	3.45
connection95374763	packaging	box,guard	18.93
connection953748889	labelling	upper_shell,serial_label	3.63
connection953749032	placement	box,parts_pack	3.45

Table 7-10: Assembly operation loaded on Wks*2*

Wks*3*

The assembly sequence for the operations loaded on workstation 3 is shown on Table 7-11. Total workstation assembly operation time – 30.39 sec

Required resources: Operator, Product testing Machine, and Ultrasonic welder

Identification code	AFC type	Mating parts	Operation time (sec)
connection953748555	ultrasonic_welding	upper_shell,lower_shell	11.55
connection953748775	testing	test_mc,lower_shell	8.95
connection953748814	labelling	upper_shell,flymo_label	3.63
connection953748937	placement	box,lower_shell	3.0
connection953749010	placement	parts_pack,gtee_label	2.63

Table 7-11: Assembly operation loaded on Wks3

Percentage loading for each workstation is shown in Figure 11.

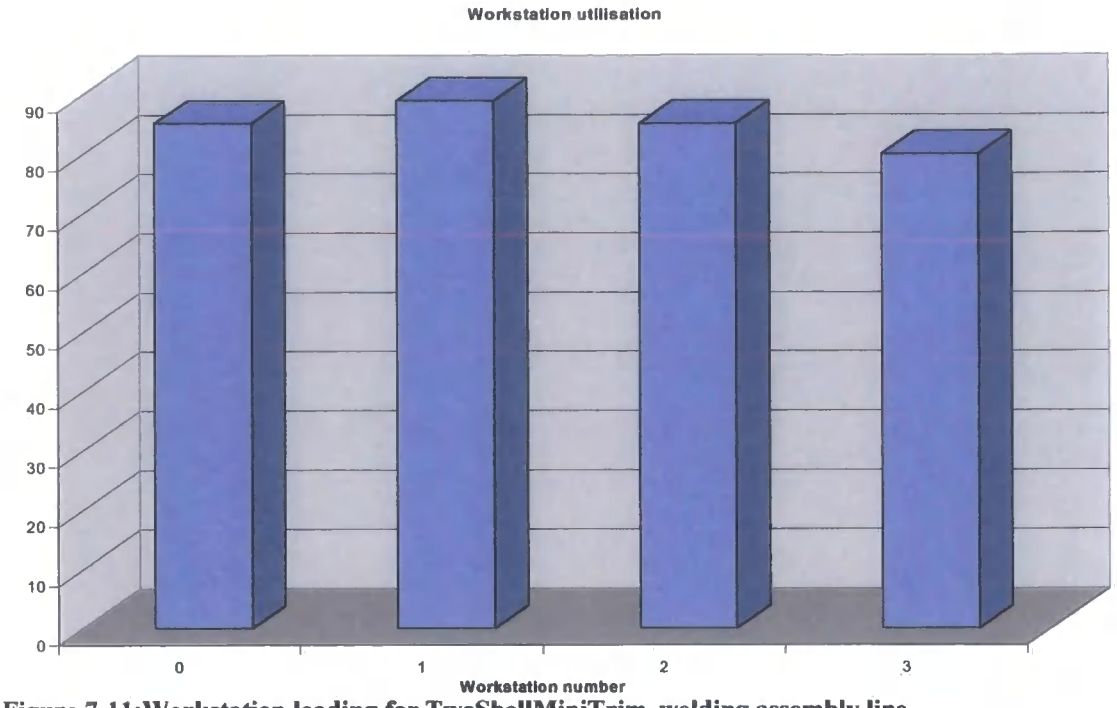


Figure 7-11:Workstation loading for TwoShellMiniTrim_welding assembly line

The results show a large degree of adherence with the optimised assembly sequence. Although the assembly sequence has been slightly altered by loading AFCs on different workstations, it is debatable whether the resulting assembly sequence is less optimised. The assembly plan generated successfully distributes the workload evenly between all workstations. It maintains a feasible assembly sequence, and a high degree of efficiency. Whilst this assembly plan may not the "best" assembly plan, it its an optimal assembly plan for the given product model based on the given assembly line layout.

7.4.2 Product model 2: TwoShellMiniTrim_screws

TwoShellMiniTrim_screws describes a low cost trimmer where the main body is formed by joining two main body moulded shells. Once all components have been assembled the main body shells are screwed together before testing and packaging processes are performed. The product model of TwoShellMiniTrim_screws is similar to that of TwoShellMiniTrim_welding, shown in Figure 7-5. Distinguishing product features of interest include:

1. 9 screws
2. 2 main body shells
3. Wiring connection between mains wire and switch assembly

The layout of the assembly line used for the analysis is shown in Figure 7-12. It is constructed as discussed in Section 6.6, for brown field assembly lines.

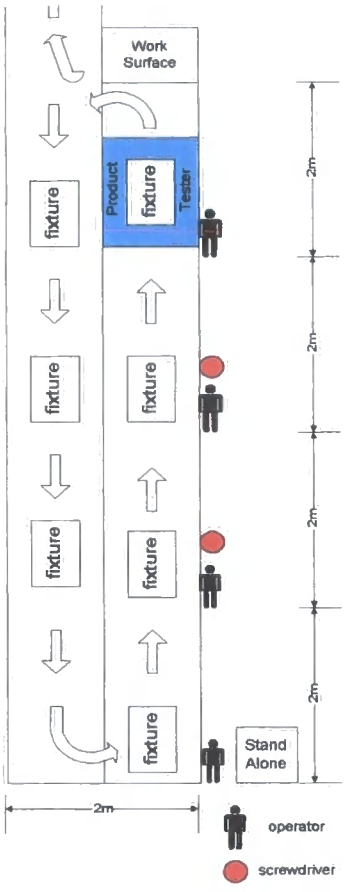


Figure 7-12: Assembly line layout for TwoShellMiniTrim_screws

As with the TwoShellMiniTrim_welding, an aggregate product model is first generated using CAPABLE*Assembly*, this model is use to create the connectivity model for TwoShellMiniTrim_screws. An initial assembly sequence is generated from the connectivity model. This sequence is used as the initial solution for generating an optimal assembly sequence using simulated annealing. The optimal assembly sequence is used to create an initial assembly plan. The generation of an optimal assembly plan is done using genetic algorithms.

The simulated annealing and genetic algorithm parameters used for the analysis are as described in Section 7.4.1. Table 7-12 shows the assembly data used to generate the optimal assembly plan using CAPABLE*Assembly*.

TwoShellMiniTrim_screws	
Factory loaded	Pseudo_Mini_Trim
Total assembly time	134.24 seconds
Maximum workstation cycle time	38 seconds
Production rate	1500
Production rate per	Day
Number of shifts	2
Length of shifts	8hrs
Number of workstations loaded	4
Number of operators	4

Table 7-12: General assembly data for TwoShellMiniTrim_screws

The optimal assembly plan for TwoShellMiniTrim_screws is presented below:

Wks*0*

The assembly sequence for the operations loaded on workstation 0 is shown on Table 7-13. Total workstation assembly operation time – 35.48 sec

Available resources: Operator, 1 stand-alone unit

Identification code	AFC type	Mating parts	Operation time (sec)
connection953722603	placement	stand_alone,cutting_head_base	2.63
connection953723062	plug_n_target	cutting_head_base,nut	3.23
connection953723168	plug_n_target	cutting_head_base, spacer.	3.23
connection953723204	plug_n_target	line_feeder,spring	2.93
connection953723250	placement	cutting_head_base,line_feeder	3.45
connection953723031	snap_fit	cutting_head_base,eye	3.83
connection953723285	placement	cutting_head_base,spool	2.63
connection953723370	snap_fit	cutting_head_base,cutting_head_cover	3.01
connection953725734	threaded	nut,axle953491634	4.46
connection953723679	placement	lower_shell,flymo_motor953491634	10.5
connection953722472	placement	fixture,lower_shell	3.45

Table 7-13: Assembly operations loaded on Wks*0*

Wks*1*

The assembly sequence for the operations loaded on workstation 1 is shown on Table 7-14. Total workstation assembly operation time – 33.66 sec

Available resources: Operator, Screwdriver

Identification code	AFC type	Mating parts	Operation time (sec)
connection953722503	wiring	flymo_switch,black_wire	3.53
connection953722552	wiring	flymo_switch,flymo_capacitor953506932	4.35
connection953723731	placement	lower_shell,switch_ass	3.45
connection953723964	plug_n_target	cable_support,orange_plug_wire	3.23
connection953723861	plug_n_target	lower_shell,cable_support	3.23
connection953722440	wiring	switch_ass,orange_plug_wire	3.42
connection953723780	wiring	flymo_motor953491634,switch_ass	5.55
connection953724050	placement	upper_shell,lower_shell	3.45

Table 7-14: Assembly operations loaded on Wks*1*

Wks*2*

The assembly sequence for the operations loaded on workstation 2 is shown on Table 7-14. Total workstation assembly operation time – 27.69 sec

Available resources: resources: Operator Screwdrivers

Identification code	AFC type	Mating parts	Operation time (sec)
connection953726017	threaded	lower_shell,screw_3	2.27
connection953725924	threaded	lower_shell,screw_5	2.27
connection953725887	threaded	lower_shell,screw_6	2.27
connection953725853	threaded.	lower_shell,screw_7	2.27
connection953725817	threaded.	lower_shell,screw_8	2.27
connection953725785	threaded	lower_shell,screw_9	2.27
connection953726097	threaded	lower_shell,screw_1	2.27
connection953725963	threaded	lower_shell,screw_4	2.27
connection953726061	threaded	lower_shell,screw_2	2.27
connection953726280	labelling	upper_shell,flymo_label	3.63
connection953726540	labelling	guard,safety_label	3.63

Table 7-15:Assembly operations loaded on Wks*2*

Wks*3*

The assembly sequence for the operations loaded on workstation 3 is shown on Table 7-16. Total workstation assembly operation time – 33.59 sec

Available resources: Operator, Screwdrivers, Product testing Machine

Identification code	AFC type	Mating parts	Operation time (sec)
connection953726203	testing	test_mc,lower_shell	20.88
connection95372868	packaging	lower_shell,Box	18.93
connection953726336	labelling	upper_shell,serial_label	3.63
connection953726610	placement	Box,guard	3.0
connection953726685	placement	parts_pack,gtee_label	2.63
connection953726714	placement	Box,parts_pack	3.45

Table 7-16: Assembly operations loaded on Wks3

Percentage loading for each workstation is shown in Figure 7-13. The balance delay for this assembly line is 14.208%, the station variation index 8.30083 seconds. This shows that although the efficiency of this assembly line is greater than the previous product model (TwoShellMiniTrim_welding) (balance delay ~19%), the workload distribution is not as high. Whilst the majority of workstations are assigned a similar workload, the operator based on Wks*1* clearly has less assembly operations assigned to him/her.

The issue of work-relatedness is more relevant in the product model. The algorithm successfully loads all threaded operations on the same workstations (Wks*2*). Thus, algorithms for multiple parts handling can be used to obtain assembly times. This decreases the time penalties imposed on the sequence of operations. This time can be further decreased if worker skill is taken into consideration. An operator working on

Wks*2*, over time, will become skilled in handling the parts in question and handling the appropriate tools. This is a practice commonly used in industry.

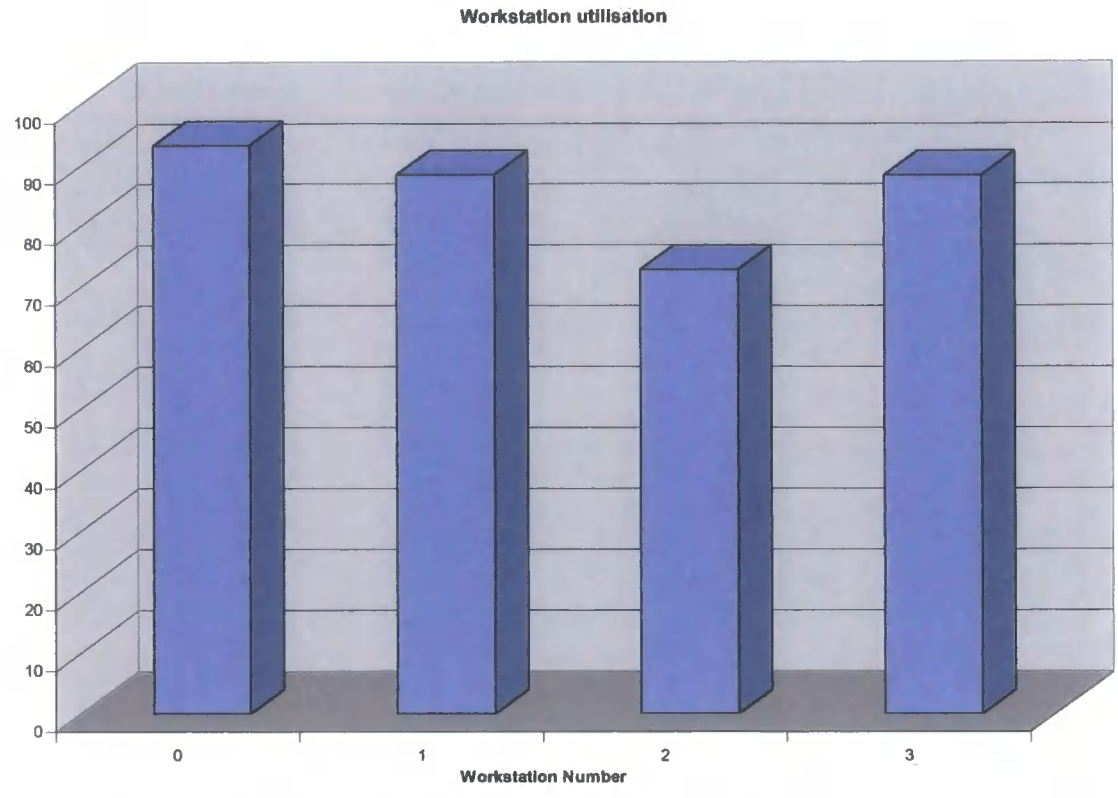


Figure 7-13: Workstation loading for TwoShellMiniTrim_screws assembly line

7.4.3 Product model 3: FourShellMiniTrim_screws

FourShellMiniTrim_screws describes a low cost trimmer where the main body is formed by joining four main body moulded shells. Once all components have been assembled the main body shells are screwed together before testing and packaging processes are performed. The product model is shown in Figure 7-14. The low cost trimmer is viewed primarily as one assembly. Distinguishing product features of interest include:

1. 9 screws
2. 4 main body shells
3. Screw type connection between mains wire and switch

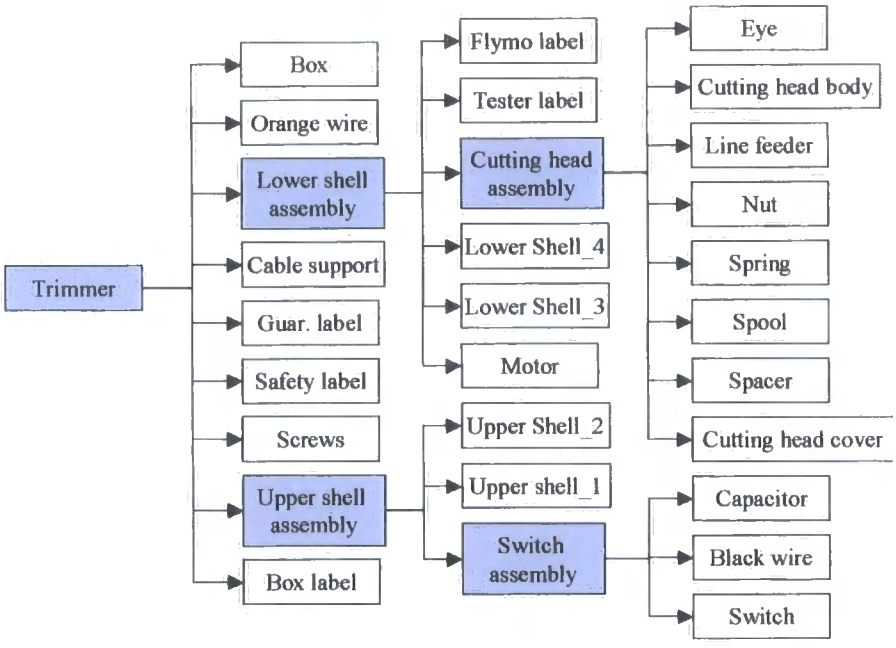


Figure 7-14:Product Model for FourShellMiniTrim_screws

The layout of the assembly line used for the analysis is shown in Figure 7-15. It is constructed as discussed in Section 6.6, for brown field assembly lines.

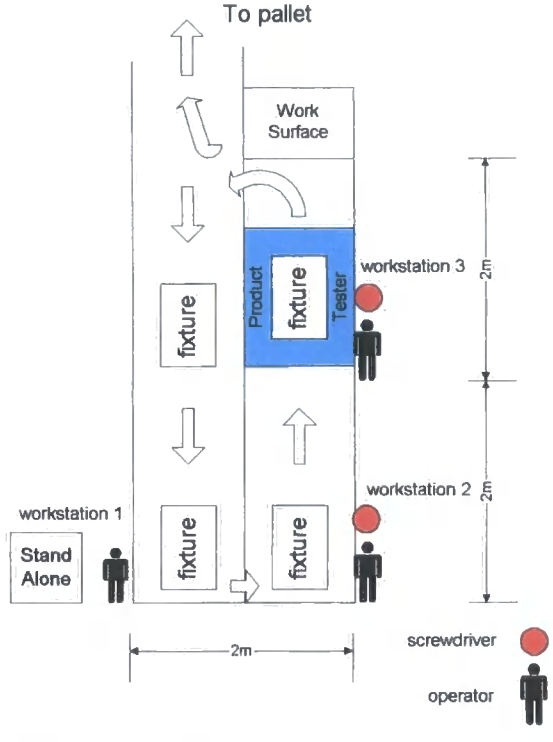


Figure 7-15:Assembly line layout for FourShellMiniTrim_screws

As before, an aggregate product model is created by extracting assembly features from the product model. The connectivity model is created using the aggregate product model; it acts as a liaison graph, providing relational and technical data for the assembly operation created. This model is used to create a feasible assembly sequence, which is optimised using a simulated annealing algorithm, with the aim of minimising the overall

assembly time. The generated optimal assembly sequence is used as the input to the genetic algorithm. The genetic algorithm is used to generate an initial assembly plan by encoding the optimised assembly sequence using genetics, and assigning workstations to each AFC (assembly operation) randomly. The system is optimised using natural selection, whereby the fittest individual in a population survives, and is passed onto/and used to create the next generation. The best individual in the final population represents the optimised assembly plan.

The simulated annealing and genetic algorithm parameters used for the analysis is as described in Section 7.4.1. Table 7-17 shows the assembly data used to generate the optimal assembly plan using CAPABLE*Assembly*.

FourShellMiniTrim_screws	
Factory loaded	Pseudo_Mini_Trim
Total assembly time	118.54 seconds
Maximum workstation cycle time	38 seconds
Production rate	1500
Production rate per	Day
Number of shifts	2
Length of shifts	8hrs
Number of workstations loaded	3
Number of operators	3

Table 7-17: General assembly data for FourShellMiniTrim_screws

Wks*0*

The assembly sequence for the operations loaded on workstation 0 is shown on Table 7-18. Total workstation assembly operation time – 37.96 sec

Available resources: Operator, 1 stand-alone unit

Identification code	**AFC type**	**Mating parts**	**Operation time (sec)**
connection953751129	placement	fixture,lower_shell_ass	2.63
connection953751282	placement	stand_alone,cutting_head_base	2.63
connection953751313	snap_fit	cutting_head_base,eye	3.83
connection953751908	plug_n_target	cutting_head_base,nut	3.23
connection953751935	plug_n_target	cutting_head_base,spacer	3.23
connection953751959	plug_n_target	line_feeder,spring	2.93
connection953751984	placement	cutting_head_base,line_feeder	3.45
connection953752009	placement	cutting_head_base,spool	2.63
connection953752036	snap_fit	cutting_head_base,cutting_head_cover	3.01
connection953752067	threaded	cutting_head_ass,flymo_motor953597243	3.71
connection953752111	placement	lower_base_shell,flymo_motor953597243	3.45
connection953752353	plug_n_target	orange_plug_wire,cable_support	3.23

Table 7-18: Assembly operations loaded on Wks*0*

Wks*1*

The assembly sequence for the operations loaded on workstation 1 is shown on Table 7-19. Total workstation assembly operation time – 35.99 sec

Available resources: Operator, Screwdriver

Identification code	AFC type	Mating parts	Operation time (sec)
connection953752491	plug_n_target	upper_base_shell,cable_support	3.23
connection953751215	wiring	flymo_switch,flymo_capacitor953598210	3.53
connection953751180	wiring	flymo_switch,black_wire	3.53
connection953751033	wiring	switch_ass,orange_plug_wire	2.27
connection953752153	placement	upper_base_shell,switch_ass	3.45
connection953752189	wiring	lower_shell_ass,upper_shell_ass	3.73
connection953753263	labelling	lower_shell_ass,flymo_label	2.63
connection953752541	placement	upper_shell_ass,lower_shell_ass	2.63
connection953752757	threaded	lower_shell_ass,screw_2	2.27
connection953752600	threaded	lower_shell_ass,screw_1	2.27
connection953753123	threaded	upper_shell_ass,screw_9	2.27
connection953753011	threaded	upper_shell_ass,screw_7	2.27
connection953753059	threaded	upper_shell_ass,screw_8	2.27
connection953752884	threaded	upper_shell_ass,screw_6	2.27

Table 7-19: Assembly operations loaded on Wks*1*

Wks *2*

The assembly sequence for the operations loaded on workstation 2 is shown on Table 7-20. Total workstation assembly operation time – 37.58 sec

Available resources: Operator, Screwdriver, Product testing Machine

Identification code	AFC type	Mating parts	Operation time (sec)
connection953752847	threaded	lower_shell_ass,screw_5	2.27
connection953752784	threaded	lower_shell_ass,screw_3	2.27
connection953752818	threaded	lower_shell_ass,screw_4	2.27
connection953753188	testing	test_mc,lower_shell_ass	15.88
connection953757688	packaging	Box, lower_shell	13.56
connection953753307	labelling	upper_shell_ass,serial_label	2.63
connection953753357	labelling	guard,safety_label	3.63
connection953753388	placement	Box,guard	3.0
connection953753450	placement	parts_pack,gtee_label	2.63
connection953753483	placement	Box,parts_pack	3.0

Table 7-20: Assembly operations loaded on Wks*2*

Percentage loading for each workstation is shown in Figure 7-16. The balance delay for this assembly line is 2.1836 %, the station variation index 1.81226 seconds. The efficiency of the assembly lines for the four shell low cost trimmer is significantly higher than both two-shell trimmers. The workload distribution is also a lot better than in the previous cases. Whilst this is a good point, such a line will be susceptible to external factors such as rejects, and human errors. The flow of assembly line can easily be disturbed, as there is little room for manoeuvre. As before, where possible, similar AFCs have been assigned to the same workstation. The assembly time on each workstation is less than the workstation cycle time.

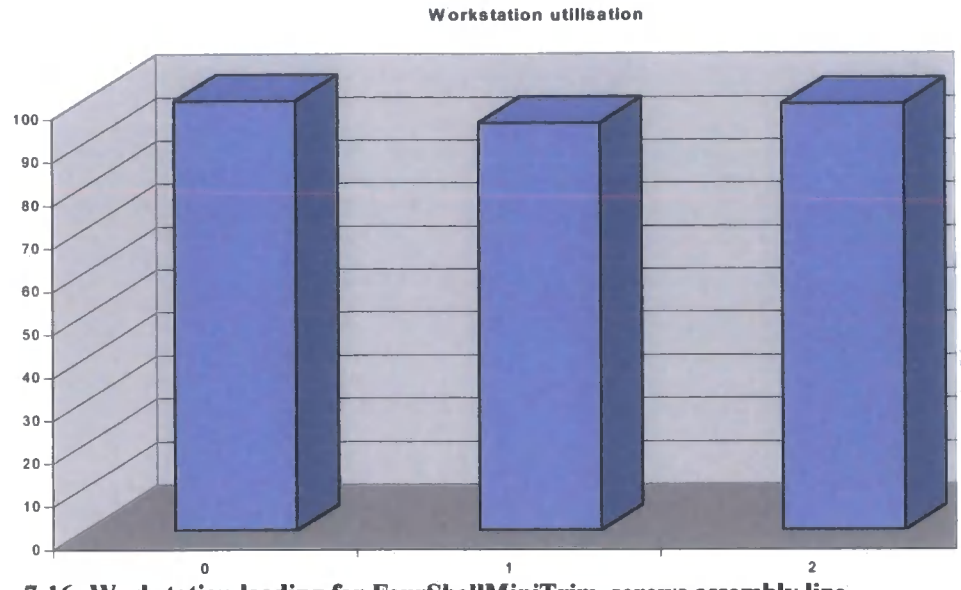


Figure 7-16: Workstation loading for FourShellMiniTrim_screws assembly line

7.4.4 Product model 4: FourShellMiniTrim_welding

FourShellMiniTrim_welding describes a low cost trimmer where the main body is formed by joining four main body moulded shells. Once all components have been assembled the main body shells are welded together before testing and packaging processes are performed. The product structure is as shown in Figure 7-14, excluding screws. The low cost trimmer is viewed primarily as one assembly. Distinguishing product features of interest include:

1. Welding features
2. 4 main body shells
3. Pressure fit connection between mains wire and switch

The layout of the assembly line used for the analysis is shown in Figure 7-17. It is constructed as discussed in Section 6.6, for brown field assembly lines.

The process of generating the optimal assembly plan follows the same route as the preceding product models. An aggregate product model is generated, creating the AFCs used to assemble the product. The connectivity model, created from the aggregate product model provides vital assembly information in terms of precedence, contact and technological data. The initial assembly sequence is derived from the connectivity model using a simple bottom-up approach. This sequence is encoded using random numbers, and is the input to the simulated annealing algorithm. This algorithm attempts to find an assembly sequence with the smallest assembly time; an optimal assembly sequence. The optimised assembly sequence is used to create an initial assembly plan.

This assembly plan is created and encoded within the genetic algorithm, which uses natural selection to find the optimal assembly plan.

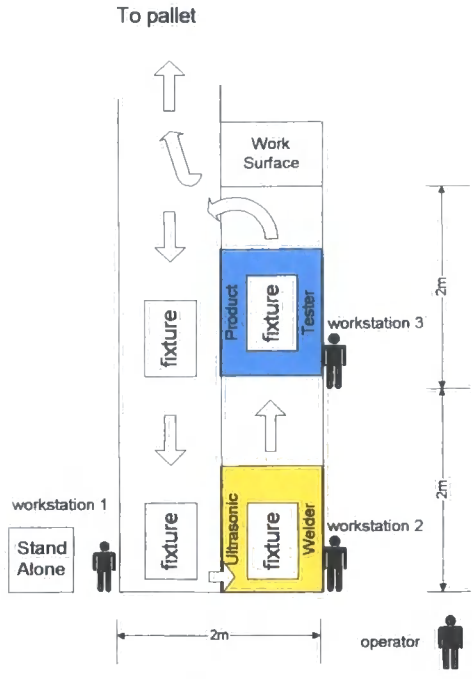


Figure 7-17:Assembly line layout for FourShellMiniTrim_welding

The simulated annealing and genetic algorithm parameters used for the analysis are as described in Section 7.4.1. Table 7-21 shows the assembly data used to generate the optimal assembly plan using CAPABLE*Assembly*.

FourShellMiniTrim_welding	
Factory loaded	Pseudo_Mini_Trim
Total assembly time	103.22 seconds
Maximum workstation cycle time	38 seconds
Production rate	1500
Production rate per	Day
Number of shifts	2
Length of shifts	8hrs
Number of workstations loaded	3
Number of operators	3

Table 7-21: General assembly data for FourShellMiniTrim_welding

Wks*0*

The assembly sequence for the operations loaded on workstation 0 is shown on Table 7-22. Total workstation assembly operation time – 35.1 sec

Available resources: Operator, 1 stand-alone unit

Identification code	AFC type	Mating parts	Operation time (sec)
connection953751282	placement	stand_alone,cutting_head_base	2.63
connection953751908	plug_n_target	cutting_head_base,nut	3.23
connection953751935	plug_n_target	cutting_head_base,spacer	3.23
connection953751959	plug_n_target	line_feeder,spring	2.93
connection953751984	placement	cutting_head_base,line_feeder	3.45
connection953751313	snap_fit	cutting_head_base,eye	3.83
connection953752009	placement	cutting_head_base,spool	2.63
connection953752036	snap_fit	cutting_head_base,cutting_head_cover	3.01
connection953752067	threaded	cutting_head_ass,flymo_motor953597243	3.71
connection953752111	placement	lower_base_shell,flymo_motor953597243	3.45
connection953751129	placement	fixture,lower_shell_ass	2.63
connection953752353	plug_n_target	orange_plug_wire,cable_support	3.23

Table 7-22: Assembly operations loaded on Wks*0*

Wks*1*

The assembly sequence for the operations loaded on workstation 1 is shown on Table 7-23. Total workstation assembly operation time – 36.0 sec

Required resources: Operator, Ultrasonic welder

Identification code	AFC type	Mating parts	Operation time (sec)
connection953752491	plug_n_target	upper_base_shell,cable_support	3.23
connection953751215	wiring	flymo_switch,flymo_capacitor953598210	3.53
connection953751180	wiring	flymo_switch,black_wire	3.53
connection953752153	placement	upper_base_shell,switch_ass	3.45
connection953751033	wiring	switch_ass,orange_plug_wire	3.53
connection953752189	wiring	lower_shell_ass,upper_shell_ass.	3.73
connection953752541	placement	upper_shell_ass,lower_shell_ass	3.45
connection953756620	ultrasonic_welding	lower_base_shell,lower_top_shell	11.55

Table 7-23: Assembly operations loaded on Wks*1*

Wks *2*

The assembly sequence for the operations loaded on workstation 2 is shown on Table 7-24. Total workstation assembly operation time – 37.4 secs

Required resources: Operator, Product testing Machine

Identification code	AFC type	Mating parts	Operation time (sec)
connection953753263	labelling	lower_shell_ass,flymo_label	2.63
connection953753188	testing	test_mc,lower_shell_ass	8.95
connection953757683	packaging	box,guard.	13.56
connection953753307	labelling	upper_shell_ass,serial_label	3.63
connection953753388	placement	box, lower_shell	3.0
connection953753450	placement	parts_pack,gtee_label	2.63
connection953753483	placement	box,parts_pack	3.0

Table 7-24: Assembly operations loaded on Wks*2*

Percentage loading for each workstation is shown in Figure 7-18. The balance delay for this assembly line is 4.8249%, the station variation index 3.08851 seconds. As with the FourShellMiniTrim_screws, the efficiency of the assembly line is less than that of both

two-shell trimmers. In this case, the workstations are not as severely loaded compared to the FourShellMiniTrim_screws. as with all other assembly plans generated, where possible similar operations have been loaded on the same workstations, operations have only been assigned to workstations with the appropriate tooling requirements, and the assembly time on each workstation is less than allowed workstation cycle time.

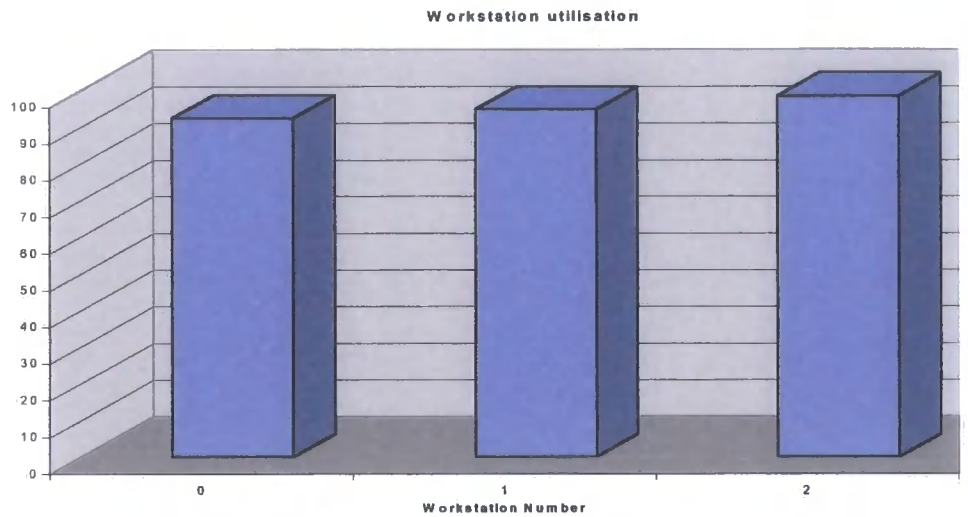


Figure 7-18:Workstation loading for FourShellMiniTrim_welding assembly line

7.4.5 Analysis of Results

The optimised assembly plans generated offer a direct comparison between the two basic designs (two shells main body and four shell main body) of the new low cost trimmer, as shown in Table 7-25.

Assembly Data	TwoShellMiniTrim (screws)	TwoShellMiniTrim (welding)	FourShellMiniTrim (welding)	FourShellMiniTrim (screws)
Production rate	1500	1500	1500	1500
Production time	Day	Day	Day	Day
Number of shifts	2	2	2	2
Number of hours	8	8	8	8
Assembly cycle time (sec)	38	38	38	38
Workstations loaded	4	4	3	3
Station variation index (sec)	8.3	7.8	3.11	1.8
Balance delay (%)	19	14	5	2
Assembly time	134.24 sec	121.47 sec	103.22 sec	118.54 sec

Table 7-25: Comparison of general assembly data

As a result of the analysis the following conclusions can be drawn:

In engineering terms:

1. Whilst the system is reasonably reliable, consistently generating feasible assembly plans with little input from the user, the system remains more tangible to persons with some knowledge of the product design. Such a user can impose

more restrictions on the movement of AFCs generated, this tends to create more consistent and reliable results. The CPU time is also reduced.

2. The simulated annealing algorithm creates an optimised assembly sequence. In an ideal world, the AFCs within the sequence would be sequentially loaded on workstations without altering the assembly sequence. This knowledge provides the user with a good idea of whether or not the optimised assembly plan is 'good'. If the assembly sequence has been severely altered, by misallocating workstations, the designer can easily spot this.

3. Using the total assembly time as a guide the assembly cost of the four shell low cost trimmer is less than that of the two shell low cost trimmer for both welding and screwing scenarios. There is approximately 18 seconds difference between the total assembly time when welded models are considered and 16 seconds difference when the low cost trimmer body shells are screwed together.

   Whilst it would be informative to perform a direct comparison with assembly times derived from the Boothroyd and/or MOST systems, technical details place it beyond the scope of this work, as such an analysis would require an adaptation of the Boothroyd and Dewhurst method on a similar scale as in this case study. This has not been undertaken due to the large extent of abstract information specific to CAPABLE*Assembly*, including relaxation values (supplied by the industrial collaborator) used in creating the assembly time database for assembly processes, as well as standard parts, and factory data. To implement a comparative Boothroyd and Dewhurst method would require some means of extrapolating such data, and mapping the estimated times and/or patterns to the Boothroyd and Dewhurst structure.

4. Welding the main body of the low cost trimmer significantly reduces the total assembly time by approximately 14 seconds (12.77 seconds in the case of two shells and 15.32 seconds in the case of four shells). The efficiency of the assembly line (balance delay) is also reduced by approximately 10 %. However, when comparing welding and screwing operations the advantage of using an ultrasonic welder is dependent on the number of screw used. This advantage is lost if less than 4 screws are used

5. The four shell low cost trimmer as opposed to the two shell low cost trimmer incurs lesser time penalties in terms of assembly, due to relatively smaller parts.

Hence part handling time and placement operation times are reduced especially where placement aids and/or guides are provided.

6. The percentage workstation loading (balance delay) of all product models are high. As the results presented represent an ideal situation (although relaxation values have been included, it is unlikely that all assembly operators will be skilled and the layout of workstations are optimised) it is safe to assume that the implementation of the assembly lines would require five workstations for the two shell low cost trimmer which resembles the current set up of the assembly lines. However, the redesign of the assembly line for the four shell low cost trimmer decreases the number of workstations required and hence, the four shell low cost trimmer will result in a reduction in the direct labour cost.

7. In general, the screw product models result in better assembly line efficiency. This is because of the smaller chunks of assembly time, which can be distributed along the assembly line. The welding models are restricted, as the entire operation needs to be performed at the same workstation.

8. The critical path lies along the path of the production of the cutting head assembly for all product models considered. In terms of redesign, a simpler cutting head with fewer components would greatly reduce the overall assembly time. While the design of the current cutting head is good in terms of general assembly as it adopts a top-down assembly direction, the number and size of components within the subassembly results in time penalties being imposed on the standard assembly operation time.

9. All assembly plans generated start by creating the cutting head assembly. If a green field assembly plan is used it, is possible to obtain assembly plan which starts by creating the switch assembly. All assembly plans generated start with the cutting head assembly because the stand-alone unit used for assembling the cutting head is situated in Wks*0*.

10. It is advisable to run the system a number of times, and compare the results of each run. Due to the nature of the problem being addressed (assembly sequence optimisation and assembly line balancing), and the optimisation methods used (simulated annealing and genetic algorithms), it is very unlikely each run will produce the same result.

In computing terms:

1. The four optimal assembly plans generated for this analysis show that CAPABLE*Assembly* is capable of producing consistently reliable results, advocating the robustness of the current system.

2. The computation time for the entire process is not high, typically, less than 15 minutes (provided an aggregate product model already exist). The majority of this time is spent reading the various files required for each module. This occurs because the system was designed such that each optimisation process can be performed separately, thus each module writes its results to a file (Microsoft Access or Excel file), which can easily be read by the next module.

3. The system will fail gracefully, and quickly if for example the aggregate product model cannot be filtered by any of optimisation modules, or an option that doesn't exist is selected. However, the system is not capable of spotting incoherent data. If the user enters invalid assembly data, the results obtained will reflect the data entered. The user will have to wait until the process is computed before any changes can be made.

4. The system has only been tested on one platform, Windows 2000. Whilst there is no reason to presume any major behavioural changes of the system, the use of standard random generators, and differences in the resolution of operating platforms might alter results obtained from the optimisation modules.

7.5 Conclusion

The fundamental methods that comprise CAPABLE*Assembly* have all been tested and validated. The results of experimental validations proved to be very encouraging. The positive feedback from the industrial case study, and the subsequent adaptation of one of the assembly process plans, show that CAPABLE*Assembly* can be used to aid assembly process planning and to some extent, product design for assembly, at the conceptual stages of design. In particular, a significant reduction in the assembly time was noted on certain product models when assembly sequences were changed to mimic those generated by CAPABLE*Assembly*. Also, altering assembly lines to use the layout found with CAPABLE*Assembly* (in terms of buffers and placement of part bins) led to an increase in the workflow. However, it was found that once an assembly line had been optimised based on a given product, the format could not be easily transferred to other product models. This suggests that work is required to generate optimal assembly plans for mixed-model assembly lines.

8 Discussion, Conclusions and Further Work

A discussion on the methods and results of the work presented is given in this Chapter. This is followed by the conclusions that can be drawn as a result of this research. To conclude, recommendations for further work required to fully exploit the potential of CAPABLE*Assembly* are subsequently presented.

8.1 Discussions

The purpose of this research is to create an intrinsically simple method for the generation of optimal assembly process plans during the conceptual stages of design. A system, CAPABLE*Assembly* was developed to demonstrate and validate the method. CAPABLE*Assembly* achieves its simplicity by adopting a 'black box' approach to generating and evaluating assembly process plans. Whilst the computational methods used within CAPABLE*Assembly*, namely simulated annealing and genetic algorithms, are tried and tested methods, the mode of implementation within CAPABLE*Assembly* presents an innovative approach to the issue of sequence generation, and assembly line balancing. In particular, defining and coding the problem to be solved using simulated annealing, and genetic algorithms is the key novel part of this research.

The method is based on three basic building blocks:

- The aggregate and connectivity product models.
- The assembly time generation methods; standard parts database and standard part assembly methodology.
- The optimisation methods; simulated annealing and genetic algorithms.

The aggregate product model is an ideal choice for modelling products at the conceptual stages of design where designers and process engineers do not yet have definitive geometrical data. The creation of a product model based on the extraction of a part's assembly features presents a simple and concise way of representing a product or component. This makes it possible to reconstruct a product based on the mating relationships (AFCs) between its constituting parts. This greatly reduces the size of the product model used for analysis, which is important in today's CAD/CAM industry for product data transfer requirements. The issue of CPU is of less importance as computers become increasingly faster.

The backbone of the aggregate product model is a feature-based solid model; this encourages a speedy assembly modelling process, and aids the generation of a

connectivity model. The connectivity model was created to certify the feasibility of the assembly sequences created from the aggregate product model, which initially could not be ensured. This is because the aggregate product model does not give adequate information in terms of relational data between mating components. For example, the aggregate product model tells you Part A is joined to Part B, but does not tell you how many other parts are linked (and how they are linked) to Parts A and B. The aggregate product model is also incapable of providing information with regards to whether assembled parts' dimensions will have an adverse effect on performing an assembly operation further down the sequence. The connectivity model is created by establishing contact, precedence and technological constraints of an assembly. This facilitates the generation of an initial rudimentary assembly plan, which not only ensures a feasible assembly plan is generated, but also reduces the search space when generating assembly sequences.

Another advantage of aggregate product modelling is that it allows for the quick modelling of products from a simple sketch or engineering drawing, making such modelling a powerful tool at the conceptual stages of design. Whilst the structure and design of the aggregate product model was presented by in part, in Betteridge (2000), here, the aggregate product model has been extended to cater for a wider range of assembly operations. Formerly, assembly operations such as packaging, labelling, and welding could not be modelled. Also, the standard parts database has been updated to reflect a wider range of motors, switches, and industry specific standard parts. The assembly times stored within the databases have been modified to reflect the work done on assembly time generation with respect to predetermined motion times (PMTS), and standard part assembly methodologies (SPAM). This vastly improved the reliability of the assembly times generated by the system when compared to assembly times achieved on the shop floor. This was one of the main hurdles that had to be overcome before the optimisation process could begin, as direct comparison with data from industry could not be made until a consistent and reliable agreement was attained. The two methods agreed to a level of 25%, which is reasonable considering other external factors not take into consideration during this research, such as the skill level of the operators, noise and other environmental factors. However, further test are required to firmly establish a more robust means of attaining assembly times.

The generated assembly plan is refined using simulated annealing. The simulated annealing algorithm seeks to generate an optimised assembly sequence by maximising

an assembly rating variable. The formulated assembly variable (Equation 5-8), in Chapter 5, Section 5.4.4, is based on the reduction of reorientations of mating components, maximisation of parallelism, and maximisation of stability of intermediate subassemblies, and is novel to this research. Each variable has been normalised to enable weighting values to be applied. This allows the system to quickly develop, and optimise assembly plans locally as well as globally. This is also novel to this research. Local optimisation occurs when at least one of the assembly variables mentioned above is ignored. Global optimisation occurs when all the assembly variables are enabled, and/or their relative importance changed. Hence, it is possible to optimise an assembly sequence based on one or more of the assembly variables. The advantage of this is that it gives the process/design engineer the freedom to investigate the assemblability of a product based on a number of criterions. He/she can make a targeted comparison, as to whether it is the stability of the intermediate subassemblies that increases the total assembly time, or if it is the fact that the products requires a lot of reorientations, and the introduction of some form of a jig to decrease the handling operations would be of benefit. This makes CAPABLE*Assembly* an effective tool for simultaneously considering several manufacturing constraints at the design stage.

CAPABLE*Assembly* runs in two modes, green and brown field. If a green field is employed, the workstations are devoid of restrictions from operator skill, tooling requirements and shop floor space. If a brown field is employed the loading of assembly operations onto workstations is limited by tool availability and operator skill. The model uses genetic algorithms to generate optimised assembly workstation loadings, employing the minimisation of cycle time, and number of workstations as the basis for the optimisation of the genetic algorithm. It aims to find the best solutions that lead to the maximum production rate and minimum workstation workload variance with maximum work-relatedness. The system has been designed through a series of trial and error experimental runs using a large number of products. The system has been balanced in such a way to give significantly higher objective scores to solutions with low station variation index and high values of kept precedence relations

The distinction between this system and other methods lies in the way the problem has been defined. It does not seek to simply produce assembly plans based on the minimisation of cycle time or number of workstations, rather it uses these parameters for the generation of the initial population. The main interest lies in the 'goodness' of the solutions/assembly plans generated. The goodness of an assembly plan refers to the

fact that the minimisation of cycle time and number of workstations takes into consideration not only work-relatedness, which advocates skill development in workers, and workload smoothness, which improves the flow and efficiency of an assembly line, but also promotes a sense of equality amongst assembly workers by distributing the workload evenly between workstations. It also uses an optimised assembly plan as a guide to loading assembly operations on workstations. The use of an optimised assembly plan provides the designer with a good visual idea as to how the assembly operations would be loaded using a green field site, and this can be used as an ideal assembly plan. Thus, the designer can quickly tell if/when a good enough solution has been attained. The use of an optimised assembly plan also reduces the search space (and computational time) of the optimisation process since more restrictions can be placed on the loading of assembly operations.

It has been argued (Nagi and Roach, 1996) that simulated annealing is better at fine-tuning a result and as such, the genetic algorithms should be used *before* simulated annealing to solve optimisation problems, This has been applied to the issue of just-in-time scheduling of multi-level assemblies. While it is true that the simulated annealing expertise lies in local optimisation, the gains in applying these methods in this context are limited. The application of these methods should be problem specific. If the aim is to produce a single optimised solution, then simulated annealing is adequate. However, if the aim is to generate a pool of optimised assembly plans (as is the case of CAPABLE*Assembly*), then simulated annealing becomes less attractive. As with the other local optimisation methods, simulated annealing works by repeatedly transforming the current solution to the next one, by the application of one move. The genetic algorithm works by keeping track of a set of feasible solutions. Furthermore, for the problem being addressed (assembly sequence generation and assembly line balancing), if the genetic algorithm were to be applied first, the assembly operations would be assigned to workstations without any idea of an optimal assembly sequence. Even if an optimal assembly sequence was to be obtained, it is still impractical to optimise each workstation locally, as this would require the operations to be independent of all assembly operations loaded on other workstations. Also, an attempt to optimise the assembly plan generated by the genetic algorithm using simulated annealing would be equally as futile. This would probably at best result in the initial assembly plan, as the local optimisation search space would probably resemble the final population of the genetic algorithm.

The use of simulated annealing for assembly line balancing would only serve to increase the complexity of CAPABLE*Assembly*, which would defeat one of the main assets of the current system. One of the keys to the successful exploitation of general-purpose heuristics is to choose the heuristic method that most naturally lends itself to the problem at hand, based on the adopted mapping method. The use of random numbers is akin to assembly sequence generation due to the way random numbers are decoded. Elements are ordered in accordance with their relative magnitude, and the overall sequence is evaluated. The modification applied to random numbers used here, lies in the use of array of objects (AFCs), identifiable by the random number stored as a property within the AFC. This simplifies the decoding of the generated sequence immensely.

If the assembly line balancing problem is to be tackled using simulated annealing, it would require the comparison of two distinct objects, the assembly operation, and the workstation, which would be difficult for the more traditional methods of encoding representation, such as tree, and binary representations, with a large chunk of CPU spent on encoding and decoding for evaluation. On the other hand, the nature of genetic algorithms offers a direct mapping to the assembly line balancing problem, without wasting time deriving complex representation schema to hold all the information required. The separate entities (genes and alleles) that constitute a genome can be directly mapped to the entities (assembly operations and workstations) that constitute an assembly line.

The results of the industrial study shows the system is capable of producing optimised assembly plans regardless of the complexity of the problem. The magnitude of assembly operations does however hamper the computational time. In general it was found that products with up to forty assembly operations has a total computational time of less than fifteen minutes. While this is quite high, the analysis was performed using a Pentium, 32MB RAM, and 6GB hard disk. The majority of this time is spent transferring product models (reading and writing files); the actual modelling time (creation of connectivity model, and optimisation using simulated annealing and genetic algorithms) in total is less than seven minutes.

For all product models tested, solutions will converge within one thousand generations. This time is further reduced if more restrictions can be imposed on the derivation of an optimal assembly sequence by using fixed AFCs. The need for a higher number of generations is increased as the number of assembly operations is increased to ensure a

global optimum has been reached and the system is not currently trapped in a local optimum. However, for large product models containing more than forty assembly operations the system has not been tested sufficiently to make any firm statements of with regards to the number of generations required for convergence or the CPU time for the total process.

8.2 Conclusions

There has been a considerable growth of interest in recent years in developing computer-aided assembly planning for mechanical and electro-mechanical products. The reasoning behind this is mainly due to the complexity of the process involved in generating optimal assembly sequences and process plans for both complex and simple products. With the trend in industry showing a marked increase in personalised features in many products today, the need for such a system is mounting.

Computer aided assembly process planning is mainly concerned with the automatic or interactive generation of feasible and cost effective assembly process plans. The results of the industrial case studies performed shows that CAPABLE*Assembly* has the potential to produce optimised assembly plans regardless of the complexity of the product at the conceptual stages of design. At the onset of this research the objectives set out to be accomplished include:

1. Developing a suitable means of representing a product model for assembly representation and sequencing. This has been achieved by the extensions made to the current aggregate product model and the fabrication of the connectivity model.

2. Standardising parts and assembly operations. This has been achieved by developing a standard assembly parts database, and by the creation of standard part assembly methodologies.

3. Deriving an effective and accurate means of estimating realistic assembly operation times. This has been achieved by the development of the assembly time generation algorithm, based of the amalgamation of two proven methods of assembly time generation, pre-determined motion times and Boothroyd and Dewhurst DFA method.

4. Generating optimal assembly sequences while satisfying assembly constraints. This has been achieved by formulating an overall assembly variable expression for the minimisation of assembly time, taking into consideration reorientation,

parallelism and stability. The optimisation process is performed using simulated annealing, and is based on maximising the overall assembly variable.

5. Generating optimal assembly process plans by mapping assembly sequences to a predefined or undefined factory model. This has been optimising the loading of assembly operation on brown or green field assembly lines/factory layout. An objective score for each assembly plan is accumulated based on workload smoothness, work-relatedness, and the optimal assembly sequence. The optimisation process applied using genetic algorithms, and is based on the maximisation of the objective score.

CAPABLE*Assembly* benefits from the following features:

1. The process of generating assembly process plans within CAPABLE*Assembly* is governed by heuristic search methods. This ensures that, at worst, a near optimal solution is obtained.
2. The final stage of optimisation (SALB) is performed using genetic algorithms, thus presenting the design engineer with a pool of near optimal/optimal assembly process plans.
3. The ease of the product modelling methods used within CAPABLE*Assembly* allows for an inherent simplicity of the system, as the generation of the relational model (connectivity model) used in both optimisation modules is derived within CAPABLE*Assembly* and not provided by the user. This encourages the user to view the system as a 'black box'.
4. The abstraction of the product model to generate an aggregate product model and connectivity model limits the size of the relational model used for optimisation. This allows the computational time for the optimisation modules to be kept at reasonable levels.
5. The heuristic methods chosen for the generation and evaluation of assembly sequences and process plans within CAPABLE*Assembly* allow for multi-criteria optimisation, hence the process plans can be evaluated in a holistic manner.

All the factors mentioned above have been used to produce CAPABLE*Assembly*; the results obtained using CAPABLE*Assembly* have been very encouraging. Indeed, a total of nine industrial products have been modelled, four of which are the conceptual product models presented in the industrial case study (Chapter 7). The remaining products were current designs, and redesigns of products. The process plans generated

to date have been tested on industrial assembly lines and in some cases yield a significant increase in the production rate. In such cases, modifications to the existing assembly lines and assembly sequences have been made to reflect some of the findings in this research.

8.3 Future work and recommendations

Whilst the industrial case study outlined in Chapter 7 advocates the use of CAPABLE*Assembly* as an effective means for generating optimal assembly process plans, it also bring to light certain shortcomings of the system as it currently stands. In particular, it demonstrates the inability of the system to cater for logistical planning and detailed economic evaluation for the process plans generated. The following recommendations for future work will have to be implemented if the full potential of CAPABLE*Assembly* is to be realised:

1. Mixed and batch model assembly lines: At present, CAPABLE*Assembly* only caters for single model assembly lines. The introduction of modules to deal with the production of two or more product models on a given assembly line can serve to strengthen the CAPABLE*Assembly*'s industrial applicability, as it is common practice within industry to produce models in batches if the models require similar sequences of processing or assembly operations. Saker and Pan (1998) provide a method for designing mixed-model assembly lines to minimise costs and idle time Automobile and truck assemblies are good examples of industries that would benefit from such analysis. This can also be done using genetic algorithms (Kim et al, 2000). The current algorithm can perhaps be modified such that each gene within a genome contains two allele-set objects, one denoting the assembly product model object, and the other the assembly operation being loaded. Here, a gene within a genome would represent a workstation.

2. Economic evaluations: Whilst an attempt has been made to include a cost evaluation on the process plans generated using CAPABLE*Assembly*, the method is largely rudimentary and has not been tested to a sufficient degree. CAPABLE*Assembly* would benefit from a knowledge base of economic analysis to enable trends to be found. For example, the industrial case study showed that unless a certain number of screws was exceeded, the use of an ultrasonic welder would not necessarily yield a higher production rate. However, the system is

incapable of ascertaining how long it would take for the ultrasonic welder to prove viable when used on a product with sufficiently large number of screws.

3. Design for logistics: Research looking into the planning for logistics support at the early stages of the assembly planning and design is desirable. This involves the determination of supply, transportation and maintenance of assembly support systems. The basic considerations within logistics planning are in the development of effective planning procedures that will include reductions in lead-times, limitation of resources, determination of critical shortages and increased flexibility. A knowledge-based economic analysis for DFA should include elements such as assembly technology selection modules, economic evaluations of alternative assembly technology considered for acquisition or replacement and general design modules including workstation selection, capacity analysis and workstation site selection.

4. Enhanced product and factory models: Further work is required to enhance assembly representation and reasoning heuristics to further speed up the analysis of more complex products. Further advantages can be gained from creating a link between the assembly modelling processes and costing methodologies, increasing the accuracy of the estimation of costing parameters such as material cost. Also, the system will benefit from providing an advanced user interface, making the system more user friendly.

5. Creation of dynamically linked libraries (dlls): CAPABLE*Assembly* has been designed in a modular format. As a DFA tool, the current modules can be compiled to create a dynamically linked library. This facilitates an easy means of increasing the functionalities within CAPABLE*Assembly*. Other modules that could be written and dynamically linked include improvements mentioned above such as a costing module, a mixed-model and two sided assembly line balancing module, and an enhanced factory modelling module. As the system is written in C++, the dlls can subsequently be attached to different user interfaces. The main benefit of this is that individual combinations of modules can be developed and distributed independently.

6. It would be useful to have a more effective means of testing the validity of the process plans generated by CAPABLE*Assembly*. As the system currently uses an optimised assembly sequence as a stating point for the generation of process plan, it is fairly easy to ascertain if an optimal or near optimal solution has been

reached. Also, the process plans generated to date have been tested on industrial assembly lines and in some cases yield a very significant increase in the production rate. However, a more academic means of checking the validity of the solutions obtained could eventually leave the simulated annealing algorithm redundant, as genetic algorithms, for balancing assembly lines does not demand an optimal solution as an initial solution; in contrast, the opposite is slightly preferred.

9 References

Amen, M., An exact method for cost-oriented assembly line balancing, International Journal of Production Economics, Vol. 64, pp. 187-195, 2000.

Anderson, E.J. and Ferris, M.C., Genetic algorithms for combinatorial optimisation, the assembly line balancing problem. ORSA Journal on Computing Vol. 6, No. 2, pp. 161-173,1994.

Baldwin, D.F., Abell, T.E., Lui, M.C.M., DeFazio, T.L., and Whitney D. E., An integrated computer aid for generating and evaluating assembly sequences for mechanical products, IEEE Trans. Robotics and Automation, Vol. 7, No. 1, pp. 78-94, 1991.

Barnes, C.J., Jared, G.E. and Swift, K.G., Evaluation of assembly sequences in an assembly-oriented design environment, Proceedings of the Institution of Mechanical Engineers, Vol. 214, Part B, pp. 89-93, 2000.

Bean, J., Genetic algorithms and random keys for sequencing and optimisation, ORSA Journal on Computing, Vol. 6, No. 2, pp 154-160, 1994.

Ben-Arieh, D. and Kramer, B., Computer-aided process planning for assembly: generation of assembly operations sequence, International Journal of Production Research, Vol. 32, No. 3, pp. 643-656, 1994.

Betteridge, M. J., 'A methodology for Aggregate Assembly Modelling and Planning', PhD Thesis, University of Durham, Durham, United Kingdom, 2000.

Boothroyd, G., and Alting, L., Design for Assembly and Disassembly, Annals of the CIRP, Vol. 41, No.2, pp. 625-634, 1992.

Boothroyd, G., Dewhurst, P., and Knight, W., Product Design for Manufacture and Assembly, Marcel Dekker, Inc., 1994.

Boothroyd, G. and Dewhurst, P., Product Design For Assembly, Designers Handbook, Boothroyd Dewhurst Inc., Kingston, 1987.

Boothroyd, G. and Fairfield, M.C., Assembly of Large Products, Annals of the CIRP, Vol. 40, No. 1, pp. 1-4, 1991.

Boothroyd, G. and Reynolds, C., Approximate Cost Estimates for Typical Turned Parts, Journal of Manufacturing Systems, Vol. 8, No. 3, pp. 189-194, 1989.

Bourjault, A. and Henrioud, J.M., Computer aided assembly process planning. Journal of Engineering Manufacture (Part B), Vol. 206, pp. 61-66, (1992).

Bowman, E.H., Assembly line balancing by linear programming, Operations research, pp. 385-389, 1960.

Bradley, H.D. and Maropoulos, P.G., 'A Concurrent Engineering Support System for the Assessment of Manufacturing Options at Early Design Stages', Proceeding of the Thirty-first International Matador Conference, pp. 485-492, 1995.

Bradley, H.D. and Maropoulos, P.G., A Concurrent Engineering Support System for the Assessment of Manufacturing Options at Early Design Stages, Proceeding of the Thirty-first International Matador Conference, pp. 485-492, 1995.

Bradley, H.D. and Maropoulos, P.G., A Relation Based Product Model for Computer Supported Early Design Assessment, Journal of Materials Processing Technology, Vol. 76, No. 1-3, pp. 88-95, 1997.

Braun, R.W. and Schiller, E.F., Computer Aided Planning and Design of Manual assembly systems, International journal of production research, Vol. 34, No. 8, pp. 2317-2333, 1996.

Brown, H.K., Martin-Vega, L.A., Thomas, J.S., and Wade, H.S., Industrial Perspective on Research Needs and Opportunities in Manufacturing Assembly, Journal of Manufacturing Systems, Vol. 14, No. 1, pp. 45-58, 1996.

Bullinger, H.J. and Ammer, E.D., Computer aided depicting of precedence diagrams: a step towards efficient planning in assembly, Computing and Industrial Engineering, Vol. 18, No. 3/4, pp. 165–169, 1984.

Caldwell, R.D., Ye, n., and Urzi, D.A., Re-engineering the product development cycle and future enhancements of computer-integrated manufacturing environment, International Journal of Computer Integrated Manufacturing, Vol. 8, No. 6, pp. 441-447, 1995.

Chao, T.H. and Sanderson, A.C., Task sequence planning with fuzzy Petri nets, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, pp. 755-768, 1995.

Chase, R.B., Survey of paced assembly lines, Industrial engineering, Vol. 6, pp. 14-18, 1974.

Daabub, A.M., and Abdalla, H.S., A Computer-based intelligent system for design for assembly, Computers and Industrial Engineering, Vol. 37, pp.111-115, 1999.

Davis, L., Job shop scheduling with genetic algorithms, Proceedings of the first international conference on genetic algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 136-140, 1985.

De Fazio, T.L. and Whitney, D.E., Simplified generation of mechanical assemblies, IEEE Journal of Robotics and Automation, Vol. 3, No. 6, pp. 640-658, 1987.

De Fazio, T.L., Abell, T.E., Amblard, G.P. and Whitney, D.E., Computer aided assembly sequence editing and choice: editing criteria, bases, rules and techniques', IEEE Intl. Conf. Robotics and Automation, pp. 416-422, 1990.

De Floriani, L. and Nagy, G., A graph model for face-to-face assembly. Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, pp. 75-78, 1990.

Delchambre, A., CAD Method for Industrial Assembly: concurrent design of products, equipment and control systems, John Wiley & Sons, 1996.

Delchambre, A., Computer-Aided Assembly Planning, Chapman & Hall, 1992.

Dewhurst, P., Design for Assembly, Maynard's Industrial Engineering Handbook, John Wiley & Sons, 2001.

Dossett, R., Computer Application of a Natural Language, Predetermined Motion Time System, Computer and Industrial Engineering Vol. 23, No. 1-4, pp. 319-322, 1992.

Evershiem, Spur, and Weil, Tool management: the present and the future, key note paper, CIRP, General Assembly, 1991.

Gen, M. and Cheng, R., Genetic algorithms and engineering design, John Wiley & Sons, INC., 1997.

Genaidy, A.M., Agrawal, A., and Mital, A., Computerised Predetermined Motion Time Systems in Manufacturing Industries, Computers and Industrial Engineering, Vol. 18, No. 4, pp. 571-584, 1990.

Gerez, S.H., Algorithms for VLSI design automation, John Wiley & Sons Ltd., 1999.

Ghosh, S., and Gagnon, R.J., A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems, International Journal of Production Research, No. 27, pp. 637-670, 1989.

Goldberg, D., and Lingle, R., Alleles, loci, and the travelling salesman problem, Proceedings International Conference of Genetic Algorithms and their Applications, 1995

Goldberg, D.E., Genetic algorithms in search, optimisation, and machine learning, Addison-Wesley, Reading, MA, 1989.

Goldberg, D.E., Korb, B., Deb, K., Messy genetic algorithms: motivation, analysis, and first results, Complex systems, No. 3, pp. 493-530, 1989.

Gottipolu, R.B. and. Gosh, K., An integrated approach to the generation of assembly sequences, International Journal of Computer Applications in Technology, Vol. 8, No. 3/4, pp.125-138, 1995.

Groover, M.P., Automation Production System and Computer Integrated Manufacturing, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

Hackman, S.T., Magazine, M.J., and Wee, T.S., Fast, effective algorithm for simple assembly line balancing problems. Operational Research, Vol. 37, pp. 916-924, 1989.

Hajek, B., Cooling schedules for optimal annealing, Mathematics of Operations Research, Vol.13, pp. 311-329 1988.

Hayes, P., and Wright, Automating Process Planning: Using Feature Relations, Journal of Manufacturing Systems, Vol. 6, No. 1, pp. 1-16, 1989.

Henrioud, J.M., Bonneville, F., and Bourjault, A., Evaluation and selection of assembly plans, Advances in Production Management Systems, pp. 489-496 1991.

Hoffman, T.R., Assembly line balancing: a set of challenging problems, International Journal of Production Research, Vol. 28, 1990, pp 1807-1815.

Homem de Mello L.S. and Desai R.S., Assembly planning for large truss structures in space, IEEE Transaction on Robotics and Automation, pp.404-407, 1990.

Homem de Mello L.S. and Sanderson A.C., A correct and complete algorithm for the generation of mechanical assembly sequences, IEEE Transactions on Robotics and Automation, Vol. 7, No. 2, pp. 228-240, 1991.

Homem de Mello L.S. and Sanderson A.C., Two criteria for the selection of assembly plans: maximizing flexibility of sequencing assembly task and minimizing the assembly time through parallel execution of assembly task, IEEE Transactions on Robotics and Automation, No. 7, No. 5, pp. 626-633, 1991b.

Homem de Mello, L.S., and Sanderson, A.C., AND/OR graph representation of assembly plans, IEEE Transactions on Robotics and Automation, No. 6, No.2, pp. 188-199, 1990.

Hu, T.C., Parallel sequencing and assembly line problems, Operations research, Vol. 9, pp. 841-849, 1961.

Huang, G.Q, Design for X —Concurrent Engineering Imperatives, Chapman and Hall, London, 1996.

Jackson, J.R., A computing procedure of a line balancing problem, Management science, Vol. 3, pp. 261-271, 1960.

Johnson, D. S., Aragon, C.R., McGeoch, L.A., and Schevon, C., 'Optimisation by simulated annealing: an experimental evaluation; part 1, graph portioning', Operations Research, 1989, Vol.13, pp. 865-892

Johnson, R.V., Optimally balancing large assembly lines with FABLE, Management Science, Vol. 34, pp 240-253, 1998.

Kanai, S., Takahashi, H., and Makino H., ASPEN: Computer-aided assembly planning and evaluation system based on pre-determined motion time standard' Annals of the CIRP, Vol. 45, pp. 35-39, January, 1996.

Kim, J.U. and Kim, Y.D., Simulated Annealing and Genetic Algorithms for Scheduling Products with Multi-level Product Structure, Computer and Operational Research, Vol.23, No. 9, pp. 857-868, 1996.

Kim, Y.J., Kim, Y.K., and Cho, Y., A heuristic-based genetic algorithm for workload smoothing in assembly lines, Computers in Operational Research, Vol.25, No.2, pp. 99-111, 1998.

Kim, Y.K., Kim, Y., and Kim, Y.J., Two-sided assembly line balancing: a genetic algorithm approach, Production Planning and Control, Vol. 11, No. 1, pp. 44-53, 2000.

Kim, Y.K., Kim, Y.J., and Kim. Y., Genetic algorithms for assembly line balancing using various objectives, Computer and Industrial Engineering, Vol. 30, No. 3, pp. 397-409, 1996.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., 'Optimisation by Simulated Annealing', Science, 1983, 220, pp. 671-680

Laperriere, L., and ElMaraghy, H. A., GAPP: A Generative Assembly Process Planner, Journal of Manufacturing Systems Vol. 15, No. 4, pp. 282-293, 1996.

Laperriere, L., and ElMaraghy, H. A., Planning of product assembly and disassembly, Annals of CIRP, Vol.41, No. 1, pp. 5-9, 1992.

Leu, Y.Y., Matheson, L.A. and Rees, L.P., Assembly line balancing using genetic algorithms with heuristics-generated initial population and multiple evaluation criteria, Decision Science, Vol. 25, pp. 581-606, 1994.

Lin, S. and Chang, T.C., An Integrated approach to automated assembly planning for three-dimensional mechanical products, International Journal of Production Research, Vol. 31, Nos. 5, pp. 1201-1227, 1993.

Ling, Z., Eng, T., Olson, W., and McLean C., (1999) 'Feature-based assembly modelling and sequence generation', Computers and Industrial Engineering, Vol.36, pp 17-33.

Ma, X., Finding the best possible solutions to simple assembly line balancing problems, Proceedings of the Institution of Mechanical Engineers, Vol.211, Part B, pp. 53-61, 1997.

Mäntylä, M and Shah, J.J., Parametric and Feature-Based CAD/CAM: concepts, techniques and applications, John Wiley and & Sons, INC, 1995

Maropoulos, P., Bradley, H., and Yao, Z., CAPABLE: An Aggregate Process Planning Tool-kit for Integrated Product Development, Journal of materials processing technology, Vol. 76, No.1-3, pp. 16-22, April 1998.

Maropoulos, P.G., A Novel Process Planning Architecture for Product-based Manufacture, Proceedings of the IMechE, Part B: Journal of Engineering Manufacture, Vol. 209, pp. 267-276, 1995.

Mascle, C and Figour, J., Methodological approach to sequences determination using the disassembly method, Proceedings of 2nd international conference computer integrated manufacturing, IEEE Robotics and Automation, pp. 483-490, 1990.

Maynard, H., Stegmerten, G., and Schwab, J., Methods Time Measurement, McGraw Hill, New York, 1948.

Milas, G., Assembly line balancing...Let's move the mystery, Industrial engineering, Vol. 22, pp. 31-36, 1990.

Miller, J. and Hoffman, R., Automatic assembly planning with fasteners, Proceedings of IEEE international conference on robotics and automation, pp. 69-74, 1989.

Miyakawa, S. and Ohashi, T., The Hitachi Assembly Evaluation Method, Proceedings International Conference on Product Design For Assembly, 1986.

Motavalli, S. and Islam, A., Multi-Criteria Assembly Sequencing, Computers and Industrial Engineering, Vol. 32, No. 4, pp. 743-751, 1997.

Nof, S.Y., Wilhem, W.E., and Warnecke, H., Industrial Assembly, Chapman & Hall 1997.

Pahl, G. and Beitz, W., Engineering Design, The Design Council, Springer, 1984.

Park, J.H. Kwon, D.G. and Chung, M.J., Framework for the evaluation and selection of assembly plans, IEEE, Proc. Intl. Conf. Industrial electronics, Control and Instrumentation (IECON), pp.1215-1221, Kobe, Japan, October, 1991.

Roach, A. and Nagi, R., A hybrid GA-SA algorithm for just-in-time scheduling of multi-level assemblies, Computers and industrial engineering, Vol. 30, No.4, pp. 1047-1060, 1996.

Rommey, B., Godard, M., Goldwasser, M. and Ramkumar, G., An efficient system for geometric assembly sequence generation and evaluation, Proceedings of the ASME International Computers in Engineering Conference, pp. 669-712, 1995.

Saker, B.R. and Pan, H.X., Integrated knowledge-based assembly sequence planning, Computers and industrial engineering, Vol. 34, No. 3, pp. 609-628, 1998.

Salveson, M.E, Assembly line balancing problem, Journal of industrial engineering, vol. 6, pp. 18-25, 1955.

Schmidt, L.C. and Jackman, J., Evaluating assembly sequences for automatic assembly systems, IIE Transactions, Vol. 27, pp. 23-31, 1995.

Scholl, A., Balancing and sequencing of assembly lines, Physica, Heidelbrg, 1995

Sediel, U.A. and Bullinger, H.J., Assembly sequence planning using operation networks, Production research: approaching the 21st Century, Taylor & Francis, London, pp. 495-503, 1991.

Senin, N., Groppetti, R. and Wallace, D.R., Concurrent assembly planning with genetic algorithms, Robotics and Computer Integrated Manufacturing, Vol. 16, pp. 65-72, 2000.

Suresh, G. and Sahu, S., 'Stochastic assembly line balancing using simulated annealing', International Journal of Production Research, 1994, 32, pp. 1801-1810.

Suresh, G., Vinod, V.V., and Sahu, S.A., A genetic algorithm for assembly line balancing, Production Planning and Control, No. 7, pp. 38-46, 1996.

Syan, C.S, Menon U, Concurrent Engineering: Concepts, Implementation and Practice, Chapman and Hall, London, 1994.

Syswerda, G., Uniform crossover in genetic algorithms, Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Publishers, Los Altos, CA, pp. 2-9, 1989.

Talbot, F.B., Patterson, J.H., and Gehrlein, W.N., A comparative evaluation of heuristic line balancing techniques, Management Science, No. 32, pp. 430-454, 1986.

Thomas, J.P., Nissanke, N., and Baker, K.D., A hierarchical Petri net framework for the representation and analysis of assembly, IEEE Transactions on Systems, Man., and Cybernetics, Vol. 12, No. 2, pp. 268-279, 1996.

Van Laarhoven, P.J., and Aarts, E, H., 'Simulated annealing theory and applications', D. Reidel Publishing, Boston, 1987

Wolter, J.D. A combinatorial analysis of enumerative data structures for assembly planning, IEEE Intl. Conf. Robotics and Automation, pp. 611-618, April, 1991.

Wolter, J.D., A constraint-based approach to planning with subassemblies, IEEE International Conference on Robotics and Automation, pp. 412-215, May 1990.

Wolter, J.D., On the automatic generation of assembly plans, Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, pp. 62-68, 1989.

Yao, Z., Bradley, H.D and Maropoulos, P.G., A Concurrent Engineering Approach for Supporting Weld Product Design at Early Stages of the Design Process, 5th

International Conference on Artificial Intelligence in Design, July 20-23, Lisbon, 641-660, 1998.

Ye, N., Banerjee, P., Banerjee A., and Dech, F., A Comparative Study of Assembly Planning in Traditional and Virtual Environments, IEEE Transactions on Systems, Man, and Cybernetics, 29(4), pp. 546-555, 1999.

Zandin, K., MOST: Work measurement systems, Dekker, New York, 1980.

Zha, X.F., Lim, S.Y.E. and Fok, S.C., Concurrent integrated design and assembly planning, Proceedings of the 4th International Conference on Robotics, Automation and Computer, Singapore, 1996.

Zha, X.F., Lim, Samuel, Y.E., and Fok, S.C, Integrated knowledge-based assembly sequence planning, International Journal of Advanced Manufacturing Technology, pp. 50-64, 1998.

Zhang B. and Zhang L., The automatic generation of mechanical assembly plans, PRICAI, pp. 668-672, 1990.

Zhang, W., Representation of assembly and automatic robot planning by Petri net, IEEE Transactions on Systems, Man., and Cybernetics, Vol. 29, No. 2, pp. 418-422, 1989.

Appendix A: Boothroyd and Dewhurst DFA time standards

The classification system for manual handling processes, along with its associated definitions and corresponding time standards, is presented in the following pages. The classification consists of two digits; each digit is assigned one of ten numerical symbols (0 to 9). The first digit of the coding system is divided into the following four main groups:

I.	First digit of 0-3	Parts of nominal size and weight that are easy to grasp and manipulate with one hands (without the aid of tools)
II.	First digit of 4-7	Parts that require grasping tools to handle because of their small size
III.	First digit of 8	Parts that severely nest or tangle
IV.	First digit of 9	Parts that require two hands, two persons, or mechanical assistance in handling

Groups I and II are further subdivided into categories representing the amount of orientation required, based on the symmetry of the part.

The second digit of the handling code is based on flexibility, slipperiness, stickiness, fragility, and nesting of a part. The second digit also depends on the group divisions of the first digit in the following manner:

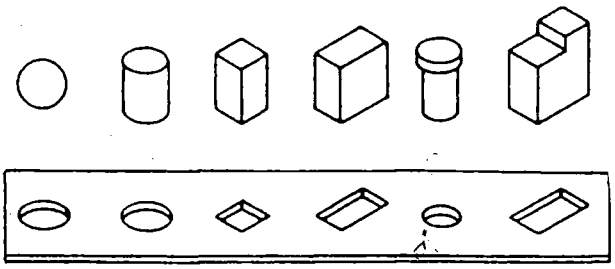
I.	First digit of 0-3	The second digit classifies the size and thickness of the part.
II.	First digit of 4-7	The second digit classifies the part thickness, type of tool required for handling the part, and the necessity for optical magnification during the handling process.
III.	First digit of 8	The second digit classifies the size and symmetry of a part.
IV.	First digit of 9	The second digit classifies the symmetry, weight, and interlocking characteristics of parts in bulk.

Key:

ONE HAND

		Parts are easy to grasp and manipulate					Parts present handling difficulties				
		Thickness > 2 mm			Thickness ≤ 2 mm		Thickness > 2 mm			Thickness ≤ 2 mm	
		Size >15 mm	6 mm ≤ Size ≤15 mm	Size <6 mm	Size >6 mm	Size ≤6 mm	Size >15 mm	6 mm ≤ Size ≤15 mm	Size <6 mm	Size >6 mm	Size ≤6 mm
		0	1	2	3	4	5	6	7	8	9
$(\alpha+\beta) < 360°$	0	1.13	1.43	1.88	1.69	2.18	1.84	2.17	2.65	2.45	2.98
$360° \le (\alpha+\beta) < 540°$	1	1.5	1.8	2.25	2.06	2.55	2.25	2.57	3.06	3	3.38
$540° \le (\alpha+\beta) < 720°$	2	1.8	2.1	2.55	2.36	2.85	2.57	2.9	3.38	3.18	3.7
$(\alpha+\beta) = 720°$	3	1.95	2.25	2.7	2.51	3	2.73	3.06	3.55	3.34	4

Parts can be grasped and manipulated by one hand without the aid of grasping tools



α	0	180	180	90	360	360
β	0	0	90	180	0	360

PART SECURED IMMEDIATELY

		No screwing operation or plastic deformation immediately after insertion (snap/press fits, circlips, spire nuts, etc.)		Plastic deformation immediately after insertion					Screw tightening immediately after insertion	
				Plastic bending or torsion			Rivetting or similar operation			
				Easy to align and position during assembly	Not easy to align or position during assembly		Easy to align and position during assembly	Not easy to align or position during assembly		
		Easy to align and position with no resistance to insertion	Not easy to align or position during assembly and/or resistance to insertion		No resistance to insertion	Resistance to insertion		No resistance to insertion	Resistance to insertion	Easy to align and position with no torsional resistance	Not easy to align or position and/or torsional resistance
		0	1	2	3	4	5	6	7	8	9
Part and associated tool (including hands) can easily reach the desired location and the tool can be operated easily	3	2	5	4	5	6	7	8	9	6	8
Part and associated tool (including hands) cannot easily reach desired location or tool cannot be operated easily. Due to obstructed access or restricted vision	4	4.5	7.5	6.5	7.5	8.5	9.5	10.5	11.5	8.5	10.5
Part and associated tool (including hands) cannot easily reach desired location or tool cannot be operated easily. Due to obstructed access and restricted vision	5	6	9	8	9	10	11	12	13	10	12

PART SECURED IMMEDIATELY

		No screwing operation or plastic deformation immediately after insertion (snap or press fits, etc.)		Plastic deformation immediately after insertion					Screwing immediately after insertion	
				Plastic bending			Riveting or similar plastic deformation			
				Easy to align and position	Not easy to align or position (no features provided for the purpose)		Easy to align and position	Not easy to align or position (no features provided for the purpose)		
		Easy to align and position no resistance to insertion	Not easy to align or position and/or resistance to insertion		No resistance to insertion	Resistance to insertion		No resistance to insertion	Resistance to insertion	Easy to align and position no resistance to screwing	Not easy to align or position and/or resistance to screwing
		0	1	2	3	4	5	6	7	8	9
Straight line insertion — From vertically above	3	1.2	1.9	1.6	2.4	3.6	0.9	1.4	2.1	0.8	1.8
Straight line insertion — Not from vertically above	4	1.3	2.1	2.1	3.2	4.8	1	1.5	2.3	1.3	2
Insertion not straight line motion	5	2.4	3.8	3.2	4.8	7.2	1.8	2.8	4.2	1.6	3.6

		Force or torque levels within robot capability								Special workhead operation
		Part can be gripped and inserted using standard gripper or gripper used for previous part				Part requires change to special gripper				
		Snap or push fit		Push and twist or other simple manipulation		Snap or push fit or simple manipulation		Screw fastening or nut running		Robot positions part
		Self-aligning	Not easy to align	Self-aligning	Not easy to align	Self-aligning	Not easy to align	Self-aligning	Not easy to align	
		0	1	2	3	4	5	6	7	8
Using motion along or about the vertical axis	3	1.0 0.55	1.0 0.6	1.0 0.7	1.0 0.75	1.0 0.6	1.0 0.65	1.0 0.7	1.0 0.8	1.0 1.15
		0 0	0 0	0 0	0 0	1.5 0.7	1.5 0.7	1.5 0.7	1.5 0.7	4.0 0.7
Using motion along or about a non-vertical axis	4	1.5 0.55	1.5 0.6	1.5 0.7	1.5 0.75	1.5 0.6	1.5 0.65	1.5 0.7	1.5 0.8	1.5 1.15
		0 0	0 0	0 0	0 0	1.5 0.7	1.5 0.7	1.5 0.7	1.5 0.7	4.0 0.7
Involving motion along or about more than one axis	5	1.5 1.05	1.5 1.1	1.5 1.15	1.5 1.2	1.5 1.05	1.5 1.1			1.5 1.6
		0 0	0 0	0 0	0 0	1.5 0.7	1.5 0.7			4.0 0.7

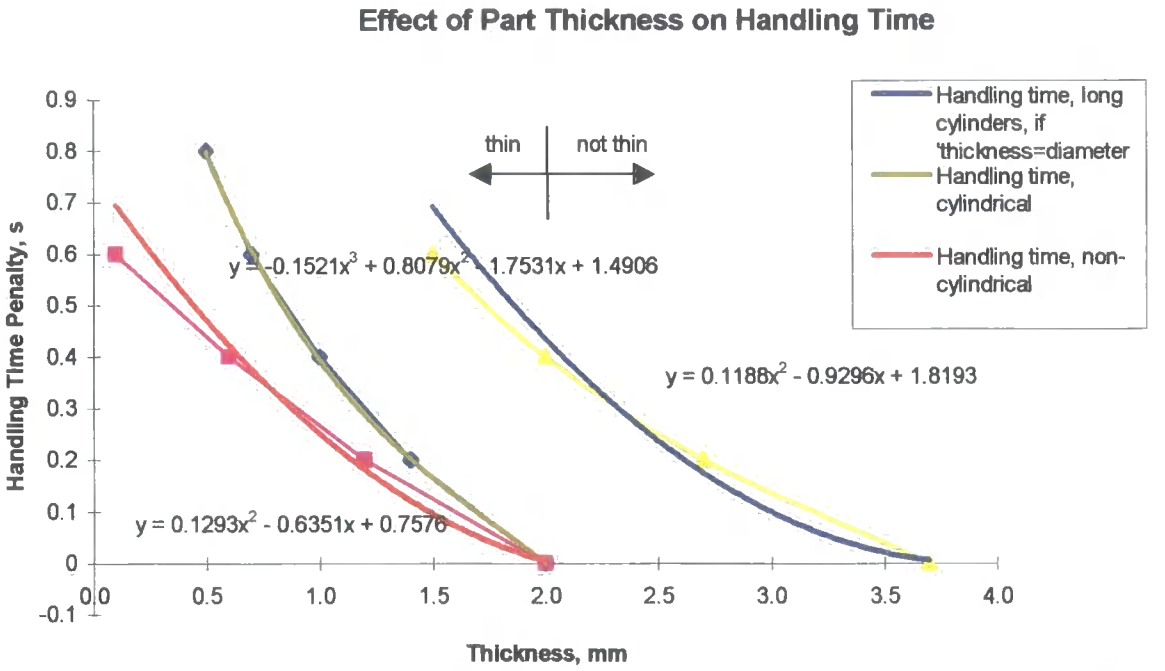
Appendix B: Graphical representation of the effects of part thickness and size on handling times.

The following data s presented in Appendix B:

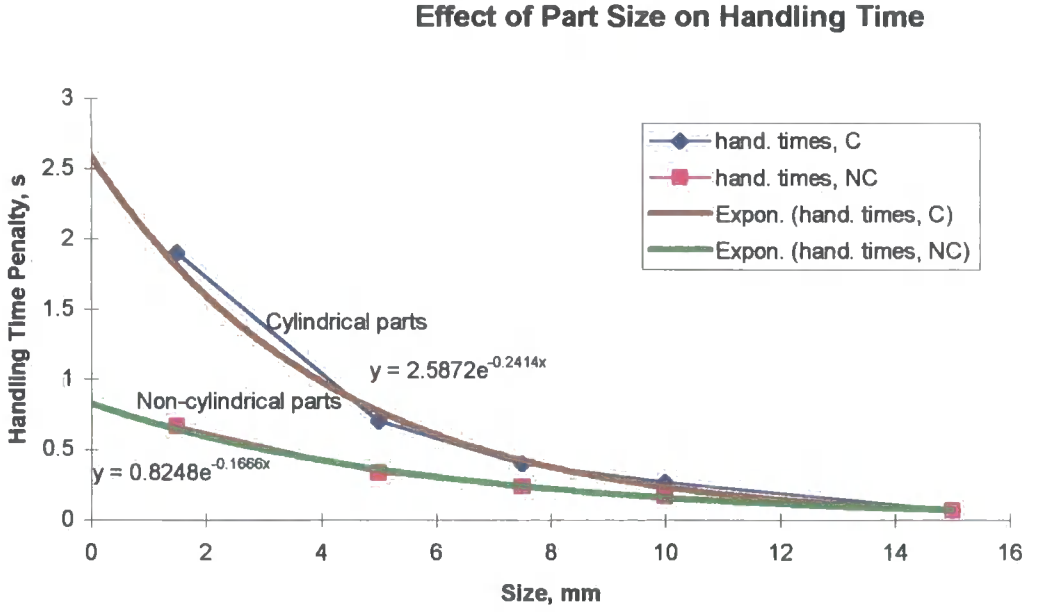
1. The effects of part thickness, and part size on handling time
2. The effect of number of threads on time taken pick-up the tool, engage the screw, tighten the screw, and replace the tool.

The effects of part thickness and size on handling time

The effect of part thickness on handling time is shown below. From the graph, it can be seen that parts with a "thickness" greater than 2mm, have no grasping or handling problems. For long cylindrical parts, this critical value occurs at 4mm.

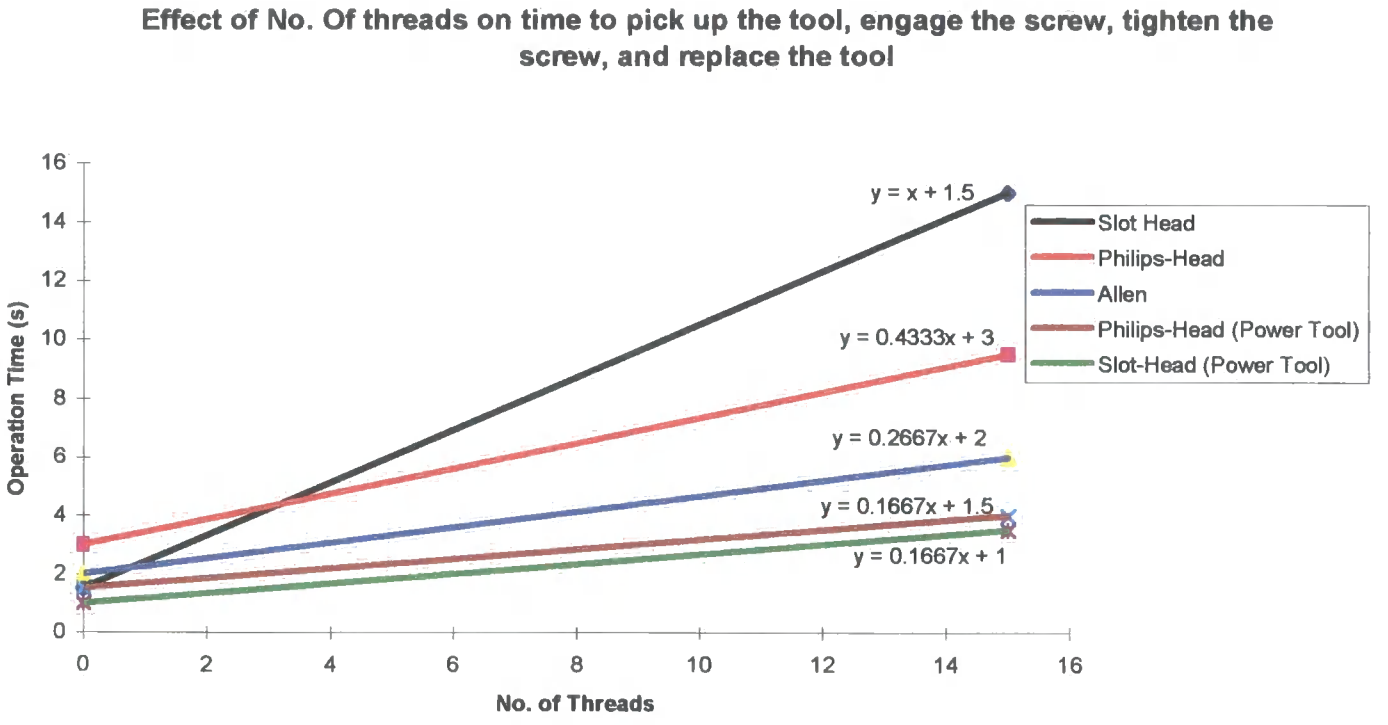


The effect of part size on handling time is shown below. From the graph, it can be seen that parts with a "size" greater than 15mm, have no grasping or handling problems. The handling time for small and medium parts shows progressively greater sensitivity with respect to part size.



The effect of number of threads on time taken pick-up the tool, engage the screw, tighten the screw, and replace the tool.

The effect of number of threads on the total time for a screwing operation, based on a variety of screw-head designs, and using both hand-operated, and power tools, is shown below. There are no restrictions on tool operation for any of the situations shown below.



Appendix C: Parameter index for assembly operations [MOST & SPAM]

The following pages show the parameter indexing currently used in MOST and SPAM

Sequence for tool Use

Guide to Parameter Indexing

A -Action Distance
Index Values
0 -Distances less than 2 inches
1 -Defined by the arc of an out streched hand

B -Body Motion
Index Values
Not considered/required at this stage.

G -Gain Control
Index Values
1 -Light objects / gain control of light objects simultaneously
3 -Gain control of heavy Tools >2.5kg

P -Placement
Index Values
1 -Lay Aside and Loose Placements
3 -Adjustments and Light Pressure Placements

F/L -Fasten/Loosen

Not considered/required at this stage.

	Obtain Tool			Place Tool					F/L	Relinquish Tool			Ass. Time		
Tool List	A	B	G	A	B	P				A	B	P	A	TMU	
Spanners															
Open	1		1	1		3				1		1		80	
Ring	1		1	1		1				1		1		60	
Adjustable	1		1	1		6				1		1		110	
Drive Systems															
Screwdrivers	1		1	1		1				1		1		60	
Wrench	1		1	1		1				1		1		60	

Power Tools															
Wrench	1		1	1		1				1		1		60	
	Obtain Tool			Place Tool							Relinquish Tool			Ass. Time	
Tool List	A	B	G	A	B	P				A	B	P	A	TMU	
Power Tools Contd.															
Slot head	1		1	1		1				1		1		60	
Philips head	1		1	1		3				1		1		80	
Screwdrivers															
Slot head	1		1	1		1				1		1		60	
Philips	1		1	1		3				1		1		80	
Allen Keys	1		1	1		3				1		1		80	
Riveting															
Lazy long type	1		3	1		3				1		1		100	
Riveting gun	1		3	1		1				1		1		80	
Manual levers	1		3	1		3				1		1		100	

Seqence for repeated tool use.

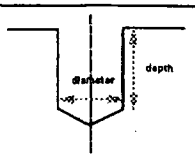
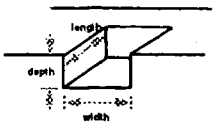
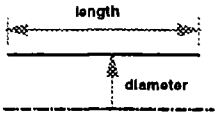
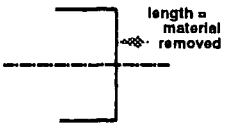
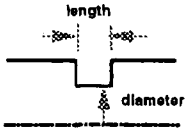
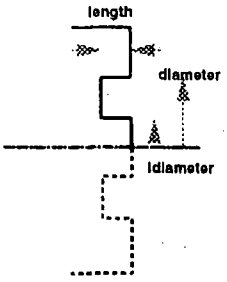
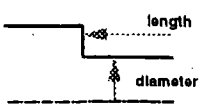
Additional guides to parameter indexing

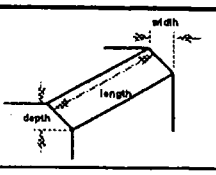
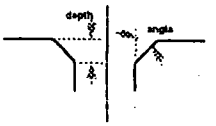
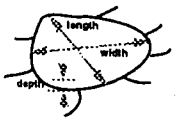
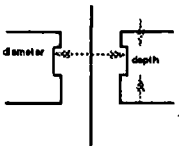
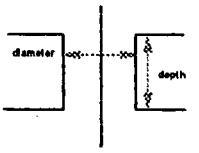
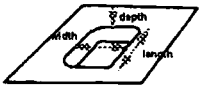
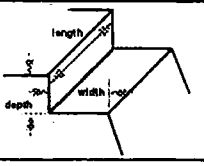
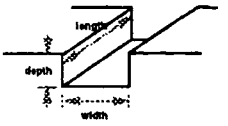
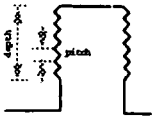
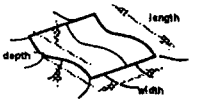
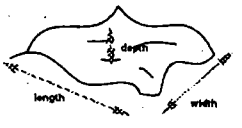
G--Gain Control
Index Values
3 -Grasp handful of parts from bin

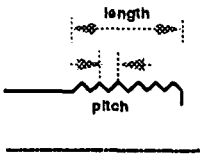
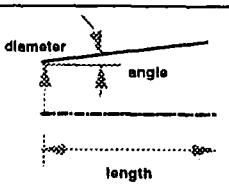
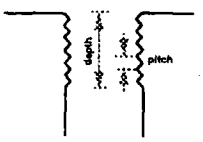
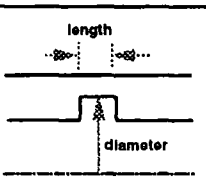
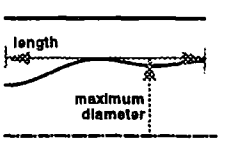
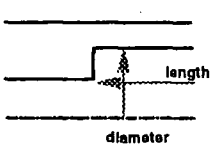
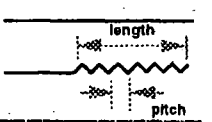
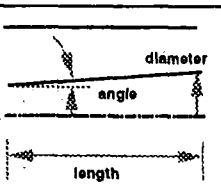
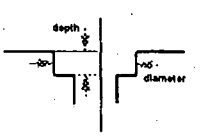
	Obtain Tool			Place Tool		Repeated Tool Use				Relinquish Tool				Ass. Time	Ass. Time
Tool List	A	B	G	A	B	**P**	**A**	**F/L**	**Parts**	A	B	P	A	TMU	secs
Spanners															
Open	1		3	1		**3**	**0**		**6**	1		1		250	8.99
Ring	1		1	1		**1**	**0**		**6**	1		1		110	3.96
Adjustable	1		1	1		**6**	**0**		**6**	1		1		410	14.75
Drive Systems															
Screwdrivers	1		1	1		**1**	**0**		**6**	1		1		110	3.96

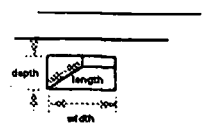
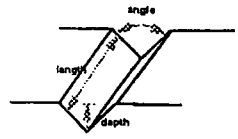
Appendix D: A comprehensive list of product features used for assembly modelling.

The following pages depict diagrammatically, the available product features used for aggregate product modelling.

Feature Class		Diagram	Parent classes	Minimum Feature Relations	Optional Feature Relations
code	description				
bho	blind hole		prismatic, holes	diameter, length	
cst	closed slot		prismatic. slots	length, width, depth	radius, angle
ecy	external cylindrical surface		axi-symmetric, external	length, diameter	
efa	end face on a cylindrical part		axi-symmetric, external, face	length, diameter, internal diameter	
egv	external groove on a cylindrical part		axi-symmetric, external,	length, diameter	
epf	external profile on a cylindrical shape		axi-symmetric, external,	length, minimum diameter	
erg	circular groove on the face of a cylindrical part		axi-symmetric, external, face	length, diameter, internal diameter	
esp	external step on a cylindrical part		axi-symmetric, external,	length, diameter	

pcf	prismatic chamfer		prismatic, face	length, width angle	
pcs	countersink: a chamfer around a hole		prismatic, hole	length, angle	
pfa	prismatic face: any flat surface		prismatic, face	length, width, depth	
pgv	cylindrical groove in a hole		prismatic, hole	length, diameter	
pho	through hole		prismatic, hole	length, diameter	
ppk	pocket		prismatic, hole	length, width, depth	
psd	shoulder on a prismatic part		prismatic, face	length, width, depth	
pst	slot		prismatic, slot	length, width, depth	radius, angle
ptd	thread on a cylindrical section of a prismatic part		prismatic, face	length, diameter, pitch	
sf2	prismatic curved surface with fixed profile		prismatic, face	length, width, depth	minimum radius
sf3	prismatic curved surface		prismatic, face	length, width, depth	minimum radius

etd	external thread on a cylinder		axi-symmetric, external,	length, diameter, pitch	
etp	external taper on a cylinder		axi-symmetric, external,	length, diameter, angle	
htd	thread on a non-axial hole		prismatic, hole	length, diameter, pitch	
icy	internal cylindrical surface on a cylindrical part		axi-symmetric, internal,	length, diameter	
igv	internal groove on a cylindrical part		axi-symmetric, internal,	length, diameter	
ipf	axi symmetrical internal profile		axi-symmetric, internal,	length, maximum diameter	
isp	internal cylindrical step		axi-symmetric, internal,	length, diameter	
itd	axi symmetrical internal thread		axi-symmetric, internal,	length, diameter, pitch	
itp	axi symmetrical internal taper		axi-symmetric, internal,	length, diameter, angle	
pcb	counterbore: a square depression around a hole		prismatic, hole	length, diameter	radius

pky	keyway		prismatic, slot	length, width, depth	
vst	v-slot		prismatic, slot	length, angle, depth	

Appendix E: Standard assembly operation times

The information presented in Appendix E, include:

1. Equations for Standard assembly operations, derived using SPAM
2. Representations of SPAM sequences

Standard assembly operations

Bolt & Nut (BN) sequences

BN_1 - $(4+6n)*\left(2.78^{(-1)}\right)+nf(N)$

BN_2 - $8n*\left(2.78^{(-1)}\right)+n\left(f(N)+g(N_e)\right)$

BN_3 - $(4+8n)*\left(2.78^{(-1)}\right)+n\left(f(N)+g(N_e)\right)$

Bolt, Nut, and Washer (BNW) sequences

BNW_1 - $12n*\left(2.78^{(-1)}\right)+nf(N)$

BNW_2 - $(8+8n)*\left(2.78^{(-1)}\right)+nf(N)$

BNW_3 - $(12+6n)*\left(2.78^{(-1)}\right)+nf(N)$

BNW_4 - $10n*\left(2.78^{(-1)}\right)+nf(N)$

BNW_5 - $(8+6n)*\left(2.78^{(-1)}\right)+nf(N)$

BNW_6 - $(12+4n)*\left(2.78^{(-1)}\right)+nf(N)$

BNW_7 - $(4+8n)*\left(27.8^{(-1)}\right)+nf(n)$

BNW_8 - $(4+10n)*\left(27.8^{(-1)}\right)+nf(n)$

Socket ratchet wrench- $y=1.7\times N$
Ring/Box end wrench $y=3\times N$
Nut Driver $y=0.8\times N$
Open-end wrench $y=3.2\times N$

N, number of revolutions / threads
y, $f(N)$

Screw (SCR) Sequences -

SCR_1 - $(4+6n)*\left(2.78^{(-1)}\right)+n\left(f(N)+g(N_e)\right)$

SCR_2 - $8n*\left(2.78^{(-1)}\right)+n\left(f(N)+g(N_e)\right)$

SCR_3 – $(4+8n)*\left(2.78^{(-1)}\right)+n\left(f(N)+g(N_e)\right)$

Allen y = 0.2667x + 2
Slot Head y = x + 1.5
Philips-Head y = 0.4333x + 3
Philips-Head (Power Tool) y = 0.1667x + 1.5
Slot-Head (Power) Tool) y = 0.1667x + 1

x, number of revolutions / threads
y, $\left(f(N)+g(N_e)\right)$

Riveting (RIV) Sequences

RIV_1 - $(10n+1)*\left(2.78^{(-1)}\right)$

RIV_2 - $(12n+1)*\left(2.78^{(-1)}\right)$

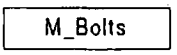
RIV_3 - $(10n+11)*\left(2.78^{(-1)}\right)$

RIV_4 - $(17n+11)*\left(2.78^{(-1)}\right)$

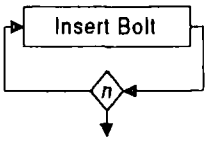
RIV_5 - $(14n+19)*\left(2.78^{(-1)}\right)$

RIV_6 - $(17n+21)*\left(2.78^{(-1)}\right)$

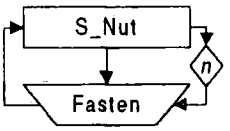
INDEX



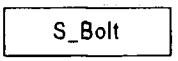
Collect a handful of bolts from a parts bin



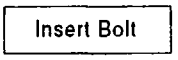
Inset a single bolt through holes. Repeat process *n* times.



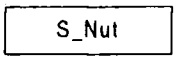
Collect a single nut form a part bin and fasten using desired tooling. Repeat process *n* times.



Collect a single bolt from a part bin.



Insert a single bolt through holes.

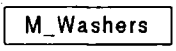


Collect a single nut from a part bin.

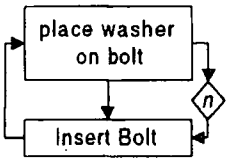
Fasten using desired tool.

Repeat the process *n* times.

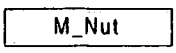
Collect a single washer from a part bin.



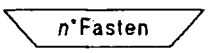
Collect a handful of washers from a part bin.



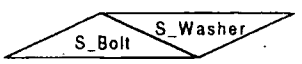
Place a single washer on a single bolt and insert bolt through holes.



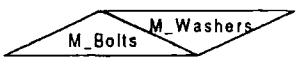
Collect a handful of nuts from a part bin.



Fasten *n* times using desired tool.

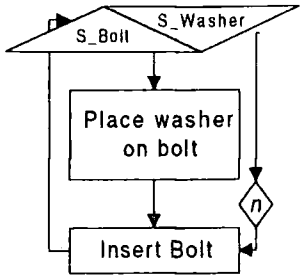


Collect a single bolt and a single washer from seperate part bins simultaneously.

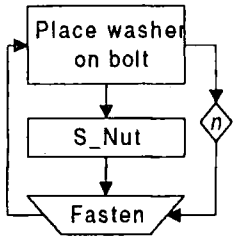


Collect a handful of bolts and a handful of washers from seperate bins simultaneously.

INDEX

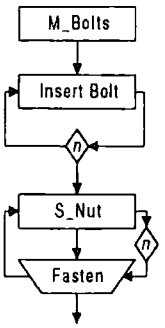


Collect a single bolt and a single washer from seperate part bins. Place the washer on the bolt and insert the bo through holes. Repeat the process *n* times



Place washer on bolt. Collect a single nut from a part bi and fasten. Repeat process *n* times.

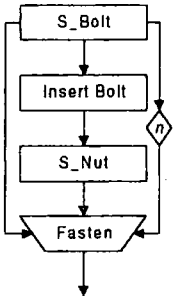
BN Systems -



BN_1 -

- Collect a handful of bolts form a part bin.
- Insert bolt through holes. Repeat process n times.
- Collect a single [1]nut from a part bin and fasten with desired tool. Repeat process n times.

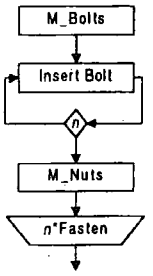
Equation: $(4+6n)*\left(2.78^{(-1)}\right)+nf(N)$



BN_2 -

- Collect a single bolt from a part bin.
- Insert bolt through holes
- Collect a single nut from a part bin.
- Bolt and nut are fastened together using desired tool.
- Process is repeated n times.

Equation: $8n*\left(2.78^{(-1)}\right)+nf(N)$



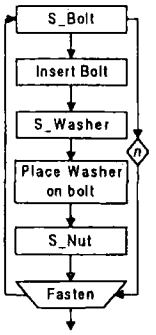
BN_3 -

- Collect a handful of bolts from a part bin.
- Insert bolt through holes. Repeat process n times.
- Collect a handful of nuts form a part bin.
- Place and fasten nuts with desired tool.

Equation: $(4+8n)*\left(2.78^{(-1)}\right)+nf(N)$

BNW Systems - P.T.O

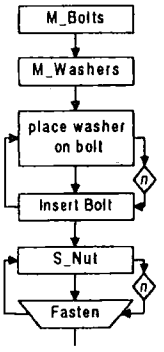
[1] The initial engagement time for placing the nut on the bolt has been included with the calculation of fastening times.



BNW_1 -

- Collect a handful of bolts from a part bin.
- Insert bolt through holes.
- Collect a single washer from a part bin.
- Place washer on bolt.
- Collect a single nut from a part bin.
- Fasten with desired tool.
- Repeat process *n* times.

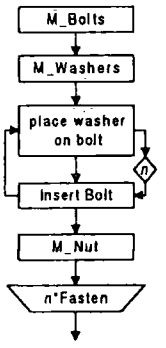
Equation: $12n * \left(2.78^{(-1)}\right) + nf(N)$



BNW_2 -

- Collect a handful of bolts from apart bin.
- Collect a handful of washers from a part bin.
- Place washer on bolt and insert bolt through holes. Repeat process *n* times.
- Collect a single nut from a part bin and fasten nut using desired tool. Repeat process *n* rimes.

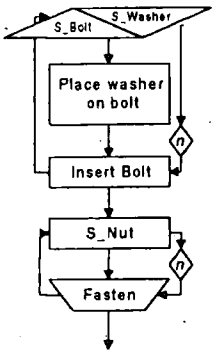
Equation: $(8 + 8n) * \left(2.78^{(-1)}\right) + nf(N)$



BNW_3 -

- Collect a handful of bolts from a part bin.
- Collect a handful of washers from a part bin.
- Place washer on bolt and insert bolt through holes. Repeat process *n* times.
- Collect a handful of nuts from a part bin.
- Place and fasten nuts using desired tool.

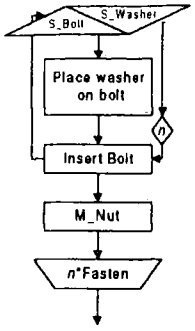
Equation: $(12 + 6n) * \left(2.78^{(-1)}\right) + nf(N)$



BNW_4 -

- Collect a single bolt and a single washer from separate part bins. Place washer on bolt and insert bolt through holes. Repeat process *n* times.
- Collect a single nut from a part bin. Fasten nut with desired tool, repeat process *n* times.

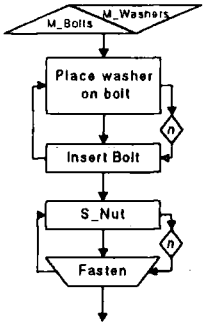
Equation: $10n * \left(2.78^{(-1)}\right) + nf(N)$



BNW_5 -

- Collect a single bolt and a single washer from separate part bins simultaneously. Place washer on bolt and insert bolt through holes. Repeat the process n times.
- Collect a handful of nuts from a part bin.
- Place and fasten nuts with desired tool.

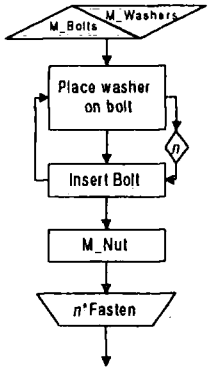
Equation: $(4+8n)*\left(2.78^{(-1)}\right)+nf(N)$



BNW_6 -

- Collect a handful of bolts and a handful of washers from separate part bins simultaneously.
- Place washer on bolt and insert bolt through holes. Repeat process n times.
- Collect a single nut from a part bin and fasten with desired tool. Repeat process n times.

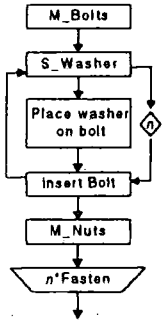
Equation: $(4+8n)*\left(2.78^{(-1)}\right)+nf(N)$



BNW_7 -

- Collect a handful of bolts and handful of washers from separate part bins simultaneously.
- Place bolt on washer and insert bolt through holes. Repeat process n times.
- Collect a handful of nuts from a part bin.
- Place and fasten nuts with desired tool.

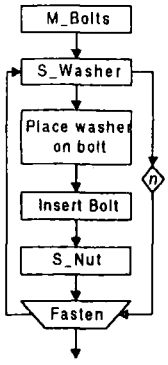
Equation: $(8+6n)*\left(2.78^{(-1)}\right)+nf(N)$



BNW_8 -

- Collect a handful of bolts from a part bin.
- Collect a single washer from a part bin and place on a bolt. Insert bolt through holes. Repeat process n times.
- Collect a handful of nuts from a part bin.
- Place and fasten nuts with desired tool.

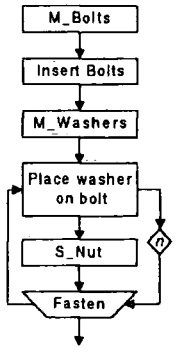
Equation: $(8+8n)*\left(2.78^{(-1)}\right)+nf(N)$



BNW_9 -

- Collect a handful of bolts from a part bin.
- Collect a single washer from a part bin. Place washer on bolt and insert bolt through holes. Collect a single nut from a part bin and fasten with desired tool.

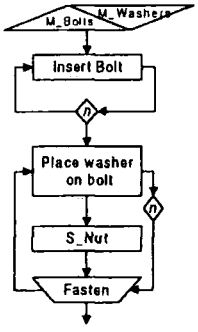
Equation: $(4+10n)*\left(2.78^{(-1)}\right)+nf(N)$



BNW_10 -

- Collect a handful of bolts from a part bin.
- Insert bolt through holes.
- Collect a handful of washers from a part bin.
- Place washer on bolt. Collect a single nut from a part bin and fasten with desired tool. Repeat process n times.

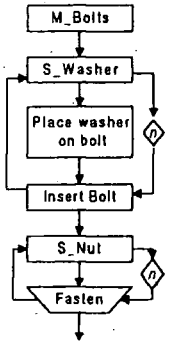
Equation: $(12+4n)*\left(2.78^{(-1)}\right)+nf(N)$



BNW_11 -

- Collect a handful of bolts and a handful of washers from separate part bins simultaneously.
- Insert bolt through holes. Repeat process n times.
- Place washer on bolt. Collect a single nut from a part bin and fasten. Repeat process n times.

Equation: $(4+8n)*\left(27.8^{(-1)}\right)+nf(n)$



BNW_12 -

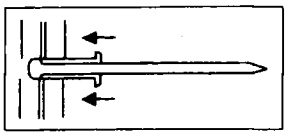
- Collect a handful of bolts from a part bin.
- Collect a single washer from a part bin and place washer on bolt. Insert bolt through holes. Repeat process n times.
- Collect a single nut from a part bin and fasten using desired tool. Repeat process n times.

Equation: $(4+10n)*\left(27.8^{(-1)}\right)+nf(n)$

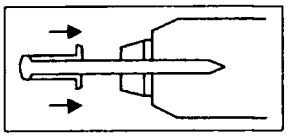
INDEX

Collect a single rivet from a part bin.

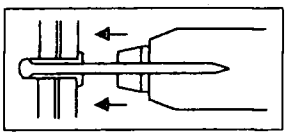
Collect a handful of rivets from a parts bin



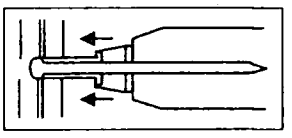
Insert a single rivet through the pre-drilled hole in the materials to be joined.



Rivet Mandrel is inserted into the riveting tool.

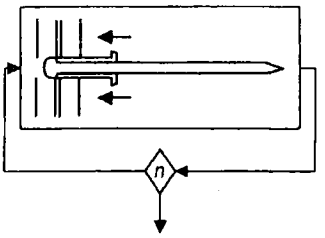


The nose piece of the riveting tool is fitted onto the rivet mandrel.of a pre-positioned rivet.

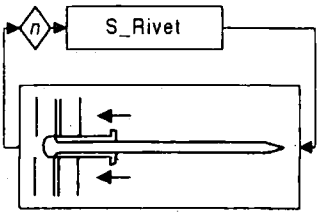


Rivet (mandrel already inserted in the riveting tool) is inserted into the pre-drilled holeof the materials to be joined.

Repeat process *n* times.

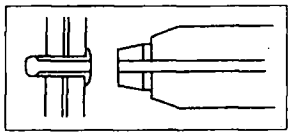


Insert a single rivet through the pre-drilled hole in the materials to be joined. Repeat the process *n* times.

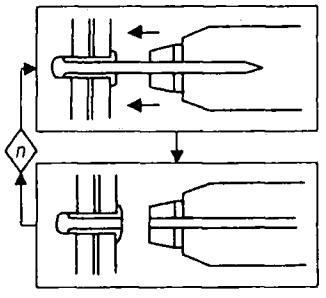


Collect a single rivet from a part bin. Insert a single rivet through the pre-drilled hole in the materials to be joined. Repeat the process *n* times.

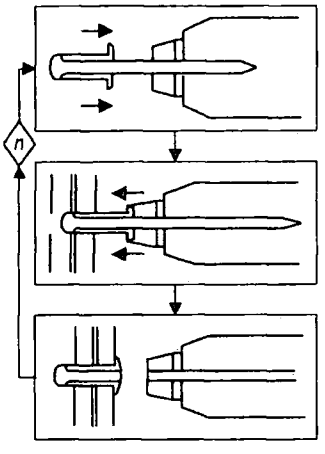
INDEX



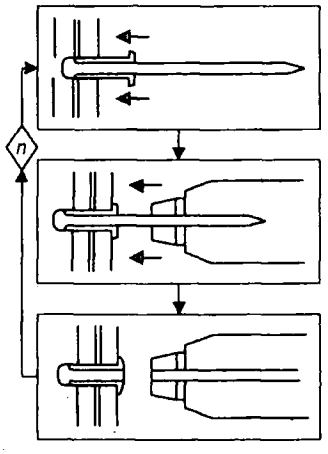
The tool is actuated (the mandrel is gripped,mandre expands and breaks at a pre-determind point).



The nose piece of the riveting tool is fitted onto the rivet mandrel.of a pre-positioned rivet. The riveting tool is actuated. Repeat the process *n* times.



Rivet Mandrel is inserted into the riveting tool. Rivet head is placed in the pre-drilled hole of the materials to be joined. The riveting tool is actuated. Repeat the process *n* times.

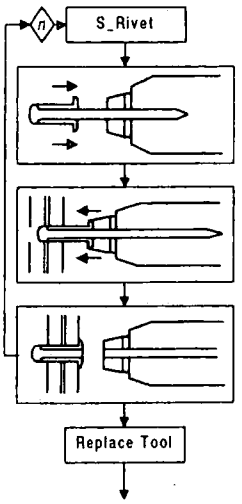


Insert a single rivet through the pre-drilled hole in the materials to be joined. Rivet mandrel is inserted in the nosse piece of the riveting tool. Riveting tool is actuated. Repeat the process *n* times.

Replace Tool

Replace riveting tool to orignal position.

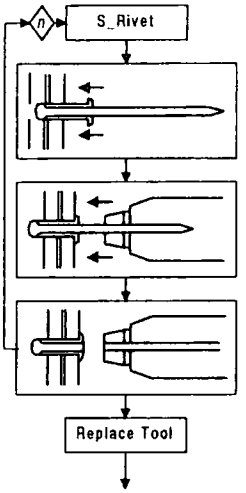
Riveting Sequences



RIV_1! -

- Collect a single rivet from a part bin
- Insert rivet in riveting tool
- Insert rivet through the pre-drilled hole in the materials to be joined.
- Actuate riveting tool.
- Repeat the process *n* times.
- Replace tool

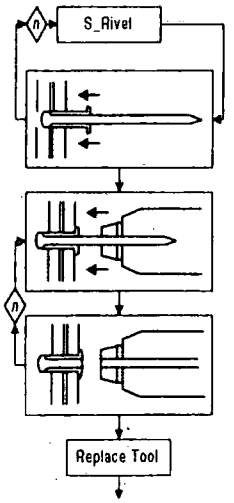
Equation: $(10n+1)*\left(2.78^{(-1)}\right)$



RIV_2 -

- Collect a single rivet from a part bin
- Insert rivet into the pre-drilled hole in the materials to be joined together.
- Insert the rivet mandrel in the riveting tool.
- Actuate riveting tool.
- Repeat process *n* times,
- Replace tool

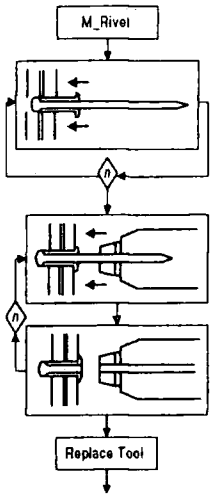
Equation: $(12n+1)*\left(2.78^{(-1)}\right)$



RIV_3 -

- Collect a single rivet from a part bin and insert into the pre-drilled hole in the materials to be joined. Repeat n times.
- Insert rivet mandrel into the nose piece of the riveting tool and actuate riveting tool. Repeat *n* times.
- Replace tool

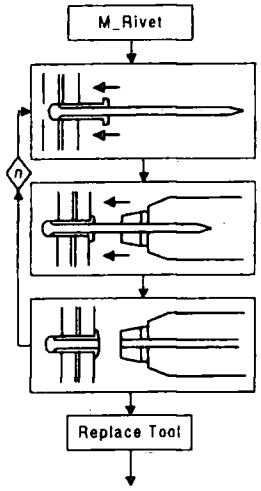
Equation: $(10n+11)*\left(2.78^{(-1)}\right)$



RIV_4 -

- Collect a handful of rivets from a part bin
- Insert rivet into the pre-drilled hole in the material to be joined. Repeat *n* times.
- Insert rivet mandrel into the nose piece of the riveting tool and actuate tool. Repeat *n* times.
- Replace tool

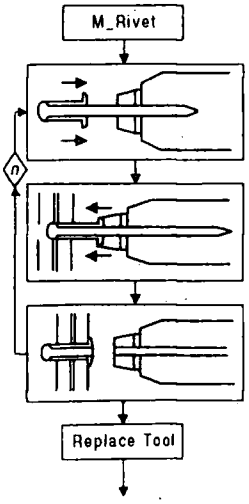
Equation: $(17n+11)*\left(2.78^{(-1)}\right)$



RIV_5 -

- Collect a handful of rivets from a part bin.
- Insert rivet into the pre-drilled hole in the material to be joined together. Insert rivet mandrel into the riveting tool and actuate tool.. Repeat *n* times
- Replace tool

Equation: $(14n+19)*\left(2.78^{(-1)}\right)$

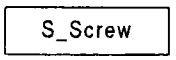


RIV_6 -

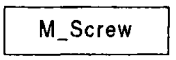
- Collect a handful of rivets from a part bin
- Insert rivet into riveting mandrel into riveting tool. Insert rivet head into the pre-drilled hole of the material to be joined and actuate tool. Repeat *n* times.
- Replace tool

Equation: $(17n+21)*\left(2.78^{(-1)}\right)$

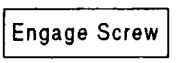
INDEX



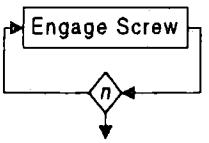
Collect a single screw from a part bin.



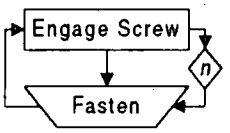
Collect a handful of screws from a parts bin



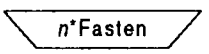
Insert a single screw through holes and engage threads with custom parts.



Engage screw threads. Repaeat the process n times.

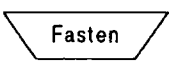


Enage screw threads with custom parts and fasten with desired tool.



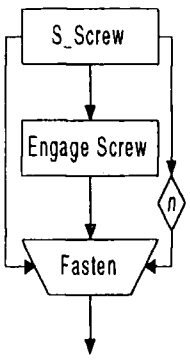
Fasten *n* times using desired tool.

Repeat the process *n* times.



Fasten using desired tool.

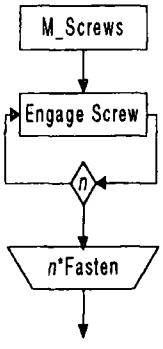
Screw Sequences -



SCR_1 -

- Collect a single screw from a part bin.
- Insert screw through holes. Engage screw thread with custom parts
- Fasten with desired tool.
- Repeat process *n* times.

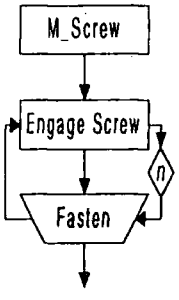
[1]Equation: $(4+6n)*\left(2.78^{(-1)}\right)+n(f(N)+g(N_e))$



SCR_2 -

- Collect a handful of screws from a part bin.
- Insert screw through holes. Engage screw threads with custom parts.
- Repeat the process n times.
- Fastened with desired tool.
- Process is repeated *n* times.

Equation: $8n*\left(2.78^{(-1)}\right)+n(f(N)+g(N_e))$



SCR_3 -

- Collect a handful of screws from a part bin.
- Insert screws through holes. Engage screw threads with custom parts and fasten with desired tool.
- Repeat *n* times

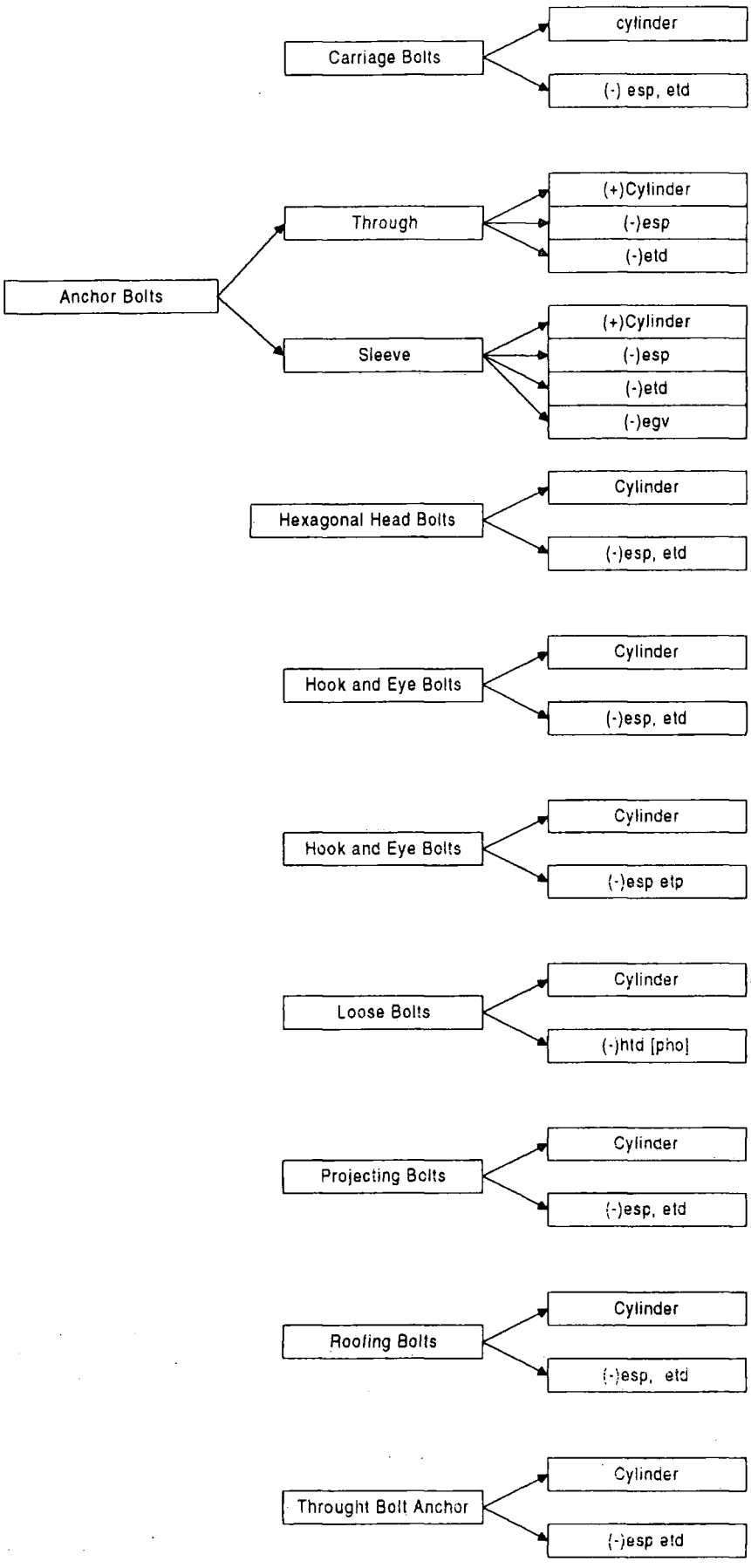
Equation: $(4+8n)*\left(2.78^{(-1)}\right)+n(f(N)+g(N_e))$

---

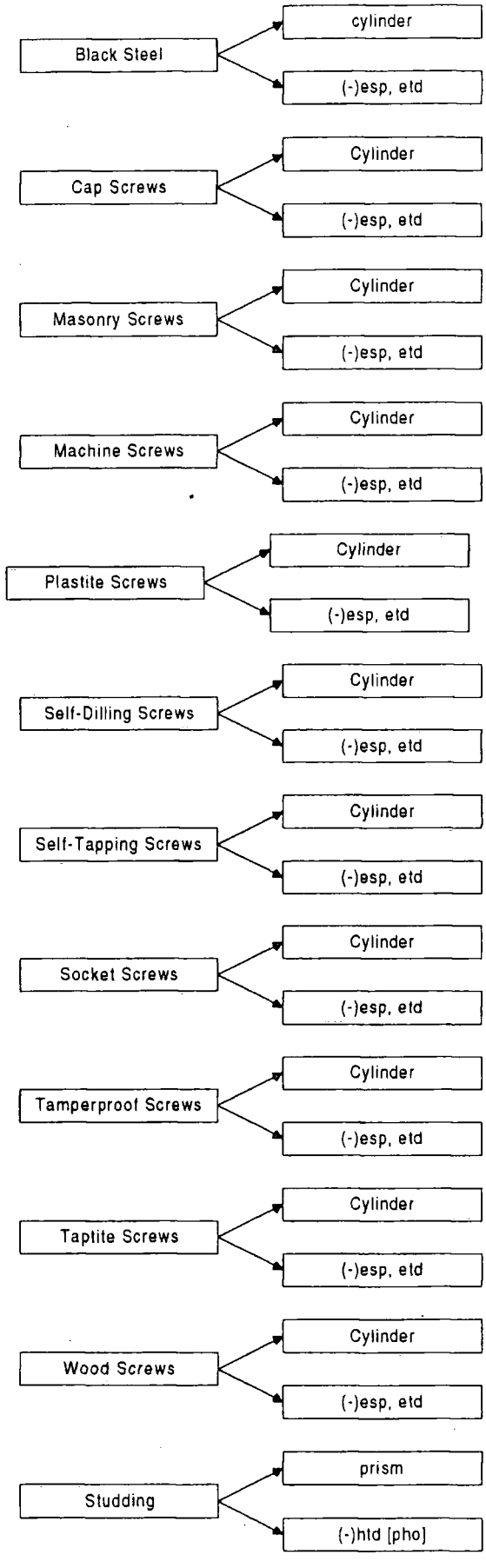
[1] $f(N)$; Function used to calculate fastening time using desired tool. Expressed as a function of time and revolutions.
$g(N_e)$; Function used to calculate engagement time that is, initial rundown time b/4 tool is applied to secure parts.

Appendix F: Comprehensive parts list of components within SPAD

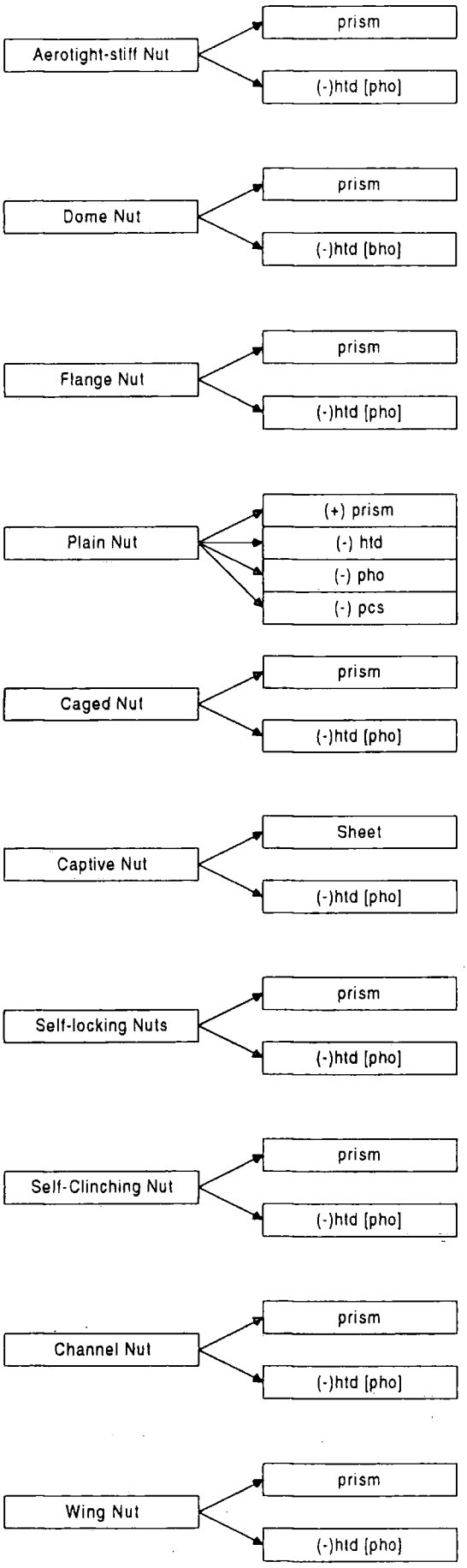
Bolts


Carriage Bolts
cylinder
(-) esp, etd
Anchor Bolts
Through
(+)Cylinder
(-)esp
(-)etd
Sleeve
(+)Cylinder
(-)esp
(-)etd
(-)egv
Hexagonal Head Bolts
Cylinder
(-)esp, etd
Hook and Eye Bolts
Cylinder
(-)esp, etd
Hook and Eye Bolts
Cylinder
(-)esp etp
Loose Bolts
Cylinder
(-)htd [pho]
Projecting Bolts
Cylinder
(-)esp, etd
Roofing Bolts
Cylinder
(-)esp, etd
Throught Bolt Anchor
Cylinder
(-)esp etd

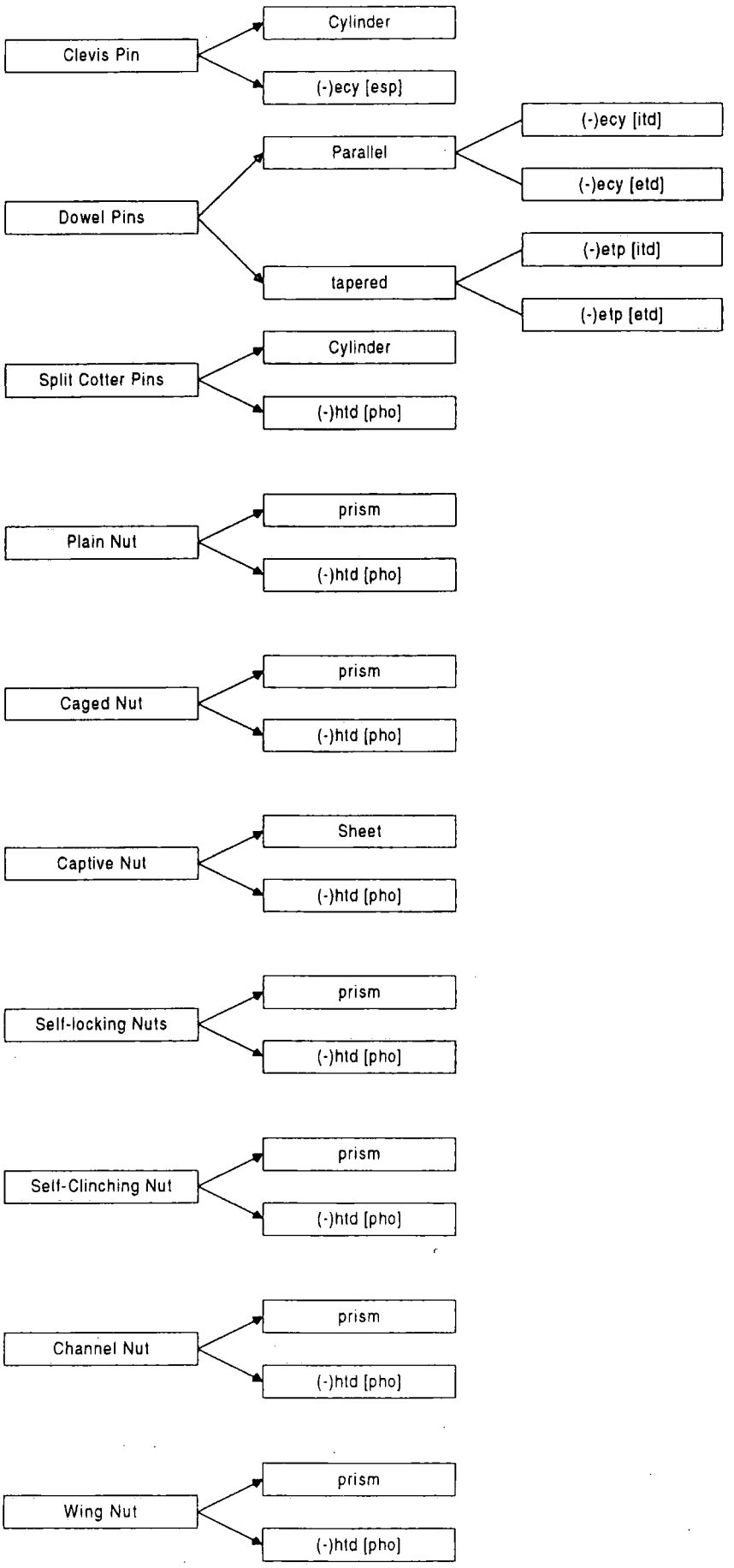
Screws & Studding



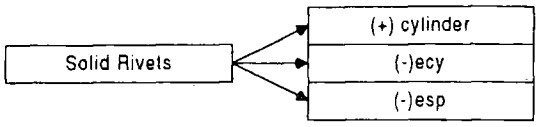
Nuts

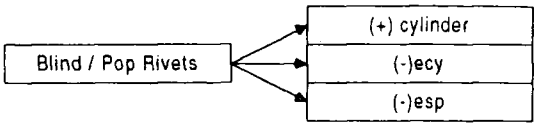

Aerotight-stiff Nut
prism
(-)htd [pho]
Dome Nut
prism
(-)htd [bho]
Flange Nut
prism
(-)htd [pho]
Plain Nut
(+) prism
(-) htd
(-) pho
(-) pcs
Caged Nut
prism
(-)htd [pho]
Captive Nut
Sheet
(-)htd [pho]
Self-locking Nuts
prism
(-)htd [pho]
Self-Clinching Nut
prism
(-)htd [pho]
Channel Nut
prism
(-)htd [pho]
Wing Nut
prism
(-)htd [pho]

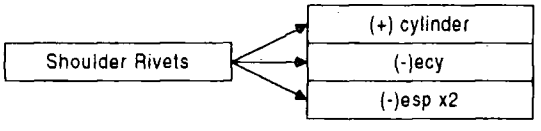
Pins

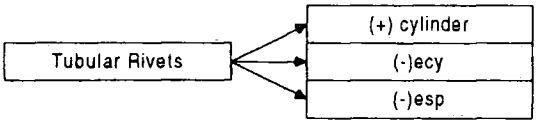

Clevis Pin
Cylinder
(-)ecy [esp]
Dowel Pins
Parallel
(-)ecy [itd]
(-)ecy [etd]
tapered
(-)etp [itd]
(-)etp [etd]
Split Cotter Pins
Cylinder
(-)htd [pho]
Plain Nut
prism
(-)htd [pho]
Caged Nut
prism
(-)htd [pho]
Captive Nut
Sheet
(-)htd [pho]
Self-locking Nuts
prism
(-)htd [pho]
Self-Clinching Nut
prism
(-)htd [pho]
Channel Nut
prism
(-)htd [pho]
Wing Nut
prism
(-)htd [pho]

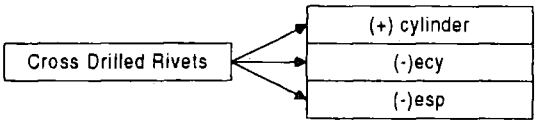
Rivets

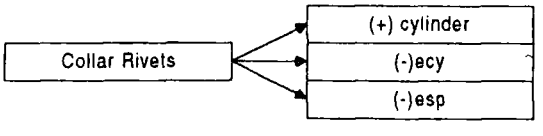

Solid Rivets
(+) cylinder
(-)ecy
(-)esp


Blind / Pop Rivets
(+) cylinder
(-)ecy
(-)esp


Shoulder Rivets
(+) cylinder
(-)ecy
(-)esp x2


Tubular Rivets
(+) cylinder
(-)ecy
(-)esp


Cross Drilled Rivets
(+) cylinder
(-)ecy
(-)esp


Collar Rivets
(+) cylinder
(-)ecy
(-)esp

Appendix G: Precedence rating (AFC ranking) of assembly operations considered

The data presented in the following page details the precedence rating and/or AFC ranking of all assembly operations considered during the course of this research.

The precedence ranking of the assembly operations (AFCs) considered within CAPABLE*Assembly* are given below. The ranking of an AFC type/sub-type denotes the preferred order of AFCs where technologically possible, within a given assembly level.

AFCs Classifications	**Assembly Feature Connections**	**Assembly Feature Connections Sub-Type**	**AFC Ranking**
Standard Insertion Assembly Operations	Placement		1
	Plug 'n' Target	Non-Cylindrical	2
		Cylindrical	3
Reversible Insertion Assembly operations	Packaging		15
	Pressure Fits		9
	Threaded	Screwing (Power tool)	7
		Bolting	8
	Wiring	Screw connectors	4
		Tag connectors	5
		Pressure-fit connectors	6
Permanent Insertion Assembly Operations	Adhesives	Gluing	13
		Labelling	14
	Riveted	Power tool (Air gun)	10
		Manual (Lever)	11
	Thermoplastic Welding	Ultrasonic welding	12
		Spin welding	12
		Hot plate welding	12
		Vibration welding	12

To calculate the AFC ranking for each AFC type a total of six products supplied by the industrial collaborator was used. The original assembly process used by the collaborator to build each product is initially broken down into assembly levels, and assembly processes that represent AFCs considered with CAPABLE*Assembly*. This essentially creates the connectivity model of each product. The relative positions of each AFC type within corresponding assembly levels for each product was then established. This information was used to infer an average order of precedence for each AFC type at each assembly level for each product. A total of 15 different types of AFCs were used for this research.

When ranking the AFCs the following factors were also taken into consideration:

1. The ranking of AFCs requires both placement operations be performed prior to the threaded operation if possible.
2. Although the rankings are hard-coded in the algorithm, the flexibility of the system can be increased by altering the default ranking of chosen AFCs.
3. The ranking of sub-types are based on the relative ease of performing the reversible operation. For example, within the wiring class of AFCs the screw connector is the most easily reversible, as screwdrivers (power tool) can easily be set to reverse motion.